Data Base Management Systems

Data Base Management Systems

Proceedings of the SHARE Working Conference on
Data Base Management Systems
Montreal, Canada, July 23-27, 1973

edited by

Donald A. Jardine

Department of Computing & Information Science
Queen's University
Kingston, Ontario, Canada

1974

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • LONDON
AMERICAN ELSEVIER PUBLISHING COMPANY, INC. – NEW YORK

Library of Congress Catalog Card Number: 74 80110

North-Holland ISBN: 07204 2804 1
American Elsevier ISBN: 0444 10672 3

PUBLISHERS:
NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM
NORTH-HOLLAND PUBLISHING COMPANY, LTD. – LONDON

SOLE DISTRIBUTORS FOR THE U.S.A. AND CANADA:
AMERICAN ELSEVIER PUBLISHING COMPANY, INC.
52 VANDERBILT AVENUE, NEW YORK, N.Y. 10017

PRINTED IN THE NETHERLANDS

INTRODUCTION

Data bases originated in the early 1960's when the first working examples, properly called data base systems rather than file handling systems, were developed. Among the earliest of these was the Integrated Data Store (IDS), developed by General Electric Company and with which the name of Charles W. Bachman is so closely associated, and the formatted file systems developed for the United States Air Force and other defence agencies.

Although these were not as complex as today's data management systems, they showed the basic concepts: an attempt to separate the structure of the data from the program and thereby institute the beginnings of data independence, and the ability to set up data structures which were not strictly linear physical sequential file organizations. These made extensive use of pointers or tables to describe the data structure. The data structure was then interpreted by software interposed between the physical data structure and the application program.

Shortly after the development of IDS, CODASYL recognized that list processing capabilities would be needed if existing languages were to handle he data structures made possible by large scale directly addressable auxiliary storage. In late 1965, CODASYL established the List Processing Task Force which later became the Data Base Task Group (DBTG). This group carried out extensive investigation, codification, and publication of a number of reports on the subject of incorporating into COBOL a facility for handling complex data structures. This work was extensively influenced by both IDS and the work of Charles Bachman, and by the Associative Programming Language developed by George G. Dodd and his colleagues at the General Motors Research Laboratory. In 1969, the CODASYL DBTG produced a proposal for the inclusion of data manipulation and data definition facilities into a host language, using COBOL as the example.

During this same period, 1967 to 1970, a group from the GUIDE and SHARE organizations were working on a set of requirements for future data base management systems, the result of which was the GUIDE/SHARE Data Base Management Systems Requirements document of November 1970.

The CODASYL DBTG document and the GUIDE/SHARE Report presented two different approaches to data base architecture. The DBTG view was that of a design and a specification, whereas the GUIDE/SHARE document contained a set of requirements, many of which went substantially beyond those specified in the CODASYL report, and indeed, went substantially beyond the foreseeable commercial state of the art at that time. The CODASYL report was the subject of considerable controversy. A number of users and a few manufacturers felt that it did not provide sufficient separation of the application program from the data structure and did not, in the view of the opponents, provide a well-considered route for user migration from existing linear file handling systems. Furthermore, the CODASYL report did not consider important problems such as the role of the data base administrator and some of the technical aspects of concurrent update and the maintenance of data integrity in a data base system.

Following the production of a revised report in April 1971 by the CODASYL Data Base Task Group, the CODASYL activity was reorganized and separated into two working groups, one which is concerned with defining a suitable data definition language, and the other with defining a data manipulation language based on COBOL, together with the syntax and semantics of an application program schema or data definition. At the time of writing, the data manipulation facility had been released and was still open for comment, while the data definition language description was close to being released.

In spite of the considerable activity by influential user groups, by CODASYL itself, and by most of the hardware and software vendors, no meeting had ever been held which would bring together various interested parties to discuss the long range implications of data bases and their architecture. Although the Association of Computing Machinery, through its Special Interest Group on File Description and Translation, had held three annual conferences, these had concerned themselves mostly with research papers in the field.

The SHARE Working Conference on Data Base Management Systems was organized to bring together, on an invitation-only basis, the most informed people in the data base management area. The attendance of over one hundred people for five days of study of this important problem attests to the enthusiasm with which the conference was received. All the papers presented at the conference were invited, so as to provide a focus on certain areas of data base technology which the organizing committee felt were important at the time. These included user experience, the presentation of user requirements, discussion on problems of migration, growth, and data independence, and an opportunity for the major software and hardware vendors to reply to the attendees on their view of data bases and what they had heard during the week. Representatives from different manufacturers, from different machine users, from a variety of corporations both in North America and Europe, from the user community and the research community gathered together to discuss mutual problems in a completely non-political and non-competitive environment, and the results were most gratifying. The success of the conference was made evident by the uniform reaction of the representatives who found that their counterparts in competitive organizations had the same problems, the same concepts, the same desire for knowledge and understanding of the way which people were using data base management systems. If the conference accomplished nothing else during its five days at the Hotel Bonaventure in Montreal, it created an opportunity for people to meet, to talk, and to discuss both formally and informally their respective problems in an atmosphere of cooperation and genuine desire to see the technology progress.

The members of the organizing committee were drawn from members of SHARE Incorporated who sponsored the conference as part of SHARE's continuing responsibility for the development and dissemination of technical information on computers and their use. The committee members were:

Thomas B. Steel, Jr., Chairman,
Donald A. Jardine, Program Chairman and Proceedings Editor
Frank Kirshenbaum, Secretary
Stuart W. Trask, Local Arrangements and Transcription
Jon A. Turner, Manager. SHARE Data Base Project.

D. A. Jardine,
Kingston,
November 1973.

ACKNOWLEDGEMENTS

These proceedings of the 1973 SHARE Workshop on Data Base Management Systems incorporate written papers as provided by the authors and edited versions of verbal discussion.

The discussions on each paper and the panel presentations were tape recorded at the conference in their entirety, and were transcribed in full during the conference itself. The detail typing was carried out by staff from Sun Life Assurance Company of Canada Limited under the supervision of Stuart W. Trask of Sun Life, to whom special thanks are due for both conference local arrangements and discussion transcription.

The individual papers and edited discussions were organized and re-typed by Mrs. Heather Ventrudo and Mrs. Alma Moore of Queen's University. Without their superb help in editing, typing, and handling a myriad of administrative duties, these proceedings could not have been produced. Diagrams were prepared by D. P. Goldring.

Final production of camera-ready copy was done by Miss Debbie Foulds of The Equitable Life Assurance Society of the United States.

CONTENTS

Data Base Management Systems, D. A. Jardine (ed.).

INFORMATION MANAGEMENT SYSTEM (IMS)
A USER'S EXPERIENCE WITH EVOLUTIONARY DEVELOPMENT

A. J. BARNETT and J. A. LIGHTFOOT
Space Division
Rockwell International Corporation
Downey, California.

INTRODUCTION

The Space Division of Rockwell International has been heavily engaged in the evolutionary development of data base management systems for over 10 years. We would like to think that this is entirely due to foresight, but in reality, many of our good efforts have been prompted as much by necessity as by ingenuity. "Necessity is the mother of invention."

In 1961, the Space Division was awarded the prime contract to design and build a space vehicle that would take man to the moon and return him safely to earth, the greatest single technological achievement in mankind's history. In addition, Rockwell International was to play a major role in the development of the Saturn, the launch booster that would send the Apollo into space. Space Division was to build the second stage of the Saturn V booster and our Rocketdyne Division was to build the rocket engines used in the several stages of the launch vehicle.

This mandate was to prove nearly as big a task informationally as it was technological. Neither computer hardware nor software existed to handle the volumes of data involved in this mammoth undertaking. For the Apollo alone, there were 100,000 drawings which grew to over two million parts, and there were 75,000 open and active production orders on our shop floor at any one time. To understand the full complexity of controlling this effort, one must realize that this was by necessity done in an R&D environment where constant change is the standard operating procedure. Man was venturing into an alien environment, and new materials, new ideas, new techniques, and new methods were all part of the day to day work of thousands. The problem of controlling this change was a monumental management task.

In early 1961, shortly after award of the Apollo contract, Space Division recognized the need for an efficient and effective indentured parts list which portrayed the various levels of assembly from the detail parts on up to the final assembly of the spacecraft. It was imperative that we and NASA have a clear indication of exactly the manner in which each spacecraft was built; and each was different, each designed for a specific task or mission.

Upon review of existing applications such as this, we found that at that time each application used a manually assigned serial number for sorting; an impossible situation with over two million parts.

As a result, we developed complex search techniques using core storage as a pseudo direct access device, but maintenance of an 18 reel master file on the IBM 7010 was not the answer. Sixty percent of our data for each part was non-unique and the number of computing steps required to produce an indentured parts list was excessive.

DATA BASE MANAGEMENT REQUIREMENT

Within a short time, IBM announced the IBM 1301 Disk file, and we quickly realized the need for a generalized access method not then existing that

1. could be taught quickly to programmers,

2. was capable of processing some form of hierarchical structure, and
3. was relatively device independent and/or language independent.

This requirement was first satisfied through a joint Space Division/IBM effort to develop a piece of software called GUAM, a generalized update and access method. This software was first used on the Disk Oriented Engineering System (DOES) at Space Division on the 7010, using the 1301 Disk file.

Through the introduction of a root segment for drawing data and a dependent segment for each of the parts list entries, redundancy was completely eliminated in DOES, and the system was efficient.

Concurrently developed with DOES in the 1964-65 period were the Engineering Document Information Collection System (EDICT) and the Logistics Inventory Management System (LIMS). These two systems were designed to provide instantaneous access to data about critical parts and drawings. To operate these two systems on a 1460 using the 7770 Audio Response Unit and the 1050 remote terminal was a new piece of Space Division/IBM joint effort software called the Response and Terminal System (RATS). This RATS monitor did some of the rudimentary communications monitoring tasks of supporting multiple applications, polling terminals, interpreting messages and calling programs, and it freed the applications programmer from concern about terminal type and/or location.

However, EDICT introduced a new type of redundancy because it dealt with essentially the same documents as DOES. Approximately one-half of the record consisted of data already in DOES, and 99 percent of the drawings in EDICT were also in DOES.

DEVELOPMENT OF IMS

The advent of S/360 caused Space Division a major reprogramming task as it did for others. Our earlier on-line software experience with RATS and GUAM spurred us to develop a message handler and a new and improved access method for S/360. This new access method became DL/1 and the message handler became IMS, again through a major developmental effort by Space Division and IBM. All of the initial IMS development was performed in Space Division offices in Downey, California.

Concurrent with the development of IMS software, Space Division began converting to S/360 with the thought in mind of taking full advantage of the new IMS capabilities. The first IMS system, an on-line Production Order Location and Reporting System (POLAR) to track our 75,000 open orders, was implemented on August 14, 1968.

Mindful of the new data base handling tools of DL/1 and IMS and realizing that we must not fall into the trap of creating a multitude of unrelated systems in an attempt to rapidly take advantage of our new "toy", Space Division instituted a different concept in systems development.

EVOLUTIONARY DEVELOPMENT

Despite the moves of many toward large MIS development, the mammoth size of our data bases, the magnitude of our existing systems, and the need for a more speedy development process made this type of approach inappropriate. Instead we launched into a long range planning process that would offer us two things, a look at where we wanted to be in the next three to five years, and a modular structure that would allow development and implementation of portions of the system while others were still in work. Consequently, our first step was to develop a complete plan and system flow of what we needed to get our job done and to support our customer, NASA, in line with contract requirements. Our next step was to segment this flow into meaningful modules that could be developed independently of all others and yet support the whole when required to do so. As part of this, interfaces were outlined so that we were assured of proper system "fit" when each independent segment became a part of the total plan.

This concept offered us several advantages over a large MIS development process: user psychological benefits, ease in defining detail system requirements, the ability to more easily manage the development task, early payback, reliability in operation, and enhancement of throughput once implemented.

User Psychological Benefits - Space Division adopted a systems development philosophy which gave the user an integral part to play in developing the system. Although working in conjunction with our Management Systems Development people, the user was given the responsibility for the preparation of the system requirements. This approach made the user a part of the team, and it made him feel that the system was his. It gave him the opportunity to present his ideas, but above all, it provided us with the cooperative spirit of the user which we all know is so vitally important to the success of any system.

Ease in Defining Detail Systems Requirements - As stated above, the user was a part of the process, giving an opening into the user area that might not have otherwise been possible. They were responsible, but most of the users realized they needed professional guidance in this new area for them. This was our raison d'etre in the user's organization. With the user playing an integral role in the process, and thereby providing intimate knowledge to the effort, requirements could be developed at a more rapid pace than previously. And the result was a specification document more clearly definitive of the users' real needs and wishes.

Ability To More Easily Manage the Development Task - By segmenting our overall plan into smaller and more easily defined tasks, we were much better able to control the development environment. Each project or development module was broken down into small tasks and/or subtasks each of which could be assigned and controlled as to cost and schedule. This provided us with a clear indication of exactly where we stood at all times and gave top management "a warm feeling" about where we were going, when we were going to get there, and the fare for the trip.

Early Payback - Since our modular development philosophy allowed us to implement various portions of the overall plan without regard to other development areas, we began getting a payback from each system as it was implemented. It is important to remember that each module had been justified on its own merits on both an economic and ability to contribute basis prior to beginning modular development. Decreases in operational costs began to amortize the development dollars, and users began to experience more efficient methods for getting their job done and consequent reductions in the number of personnel needed.

In addition, implementation of our early systems under IMS provided us a method of displaying our wares. It introduced the division to on-line systems technology and its benefits, and above all, it proved we could get the job done in a new environment. It showed that we indeed did have a software tool for more easily establishing a system and maintaining and expanding it in a changing environment.

Reliability in Operation - Since our systems were developed in a modular manner, many of our data bases were constructed to stand on their own. The expandability features of DL/1 and IMS prevents this from being a major problem to us. This has provided us with significant reliability advantages over having all of our data in one base.

If one data base goes down from either software or hardware problems, there is only one user down; all others are operative since their data has remained in service. Furthermore, should a data base be lost and recovery become necessary, the recovery task is a simple one since less data must be re-processed.

Enhancement of Throughput - We have found through our experiences with DL/1 and IMS that redundancy of data within two or more data bases is not always bad. Indeed, it has been found to be more efficient in many cases than attempting to eliminate all redundancy.

In addition, as a result of our modular system and data base structure, our data bases have tended to be less complex, to contain fewer dependent segments, and be composed of fewer data elements than an MIS data base which purports to contain all of the data needed by a company to manage its affairs. Obviously, the

greater the complexity and the larger the data hierarchy the less efficient the data access, particularly in the on-line environment. We have found that as a result, our transaction time in the message region has been excellent, our terminal response time very adequate, and the cost to our users significantly reduced over the large and complex data hierarchies that we have attempted at times and others have implemented.

PHASED APPROACH TO SYSTEMS DEVELOPMENT

The Space Division began its development of its "Integrated Information System" in 1967. As was stated earlier, the initial step was the development of a long range plan which briefly outlined the system as it was conceived and standardized many of the data elements and their sizes. The illustration on the following page shows the progress that has been made to date in the execution of that plan. This illustration shows a top level information flow into and between the major subsystems and indicates the phases which were used to segment the development. A brief explanation of the modules and their evolutionary development will show the growing confidence developed at the Space Division of Rockwell International in IMS capability and reliability.

It all started with a single DL/1 batch system and has evolved to the point that now every major aspect of the division's work is being done with IMS.

Manufacturing Planning System, Phase 1 - Our first undertaking was the development of a Manufacturing Planning System. This system consisted of two major data sets - a planning parts list and a manufacturing schedule. The system had the capability of producing indentured parts lists, and scheduling the manufacturing of detail parts and assemblies through the release of production orders. This system was initially developed as a batch system using the data access mechanism, DL/1, which was to become the backbone of the IMS/360 data management facility. This initial system has since been converted to an on-line system since all the subsequent phases of our plan have been developed as on-line systems originally.

Production Control System, Phase 2 - The second phase was a production control system (Polar II) which was capable of tracking the status of approximately 60,000 active production orders. As indicated earlier, this was actually the first on-line system implemented using IMS/360 as it is known today.

The implementation of this Phase 2 system in August, 1968, provided a good test for simple IMS data base management activity as well as providing experience in the use of the communication facility.

Material Status System, Phase 3 - Immediately following our Polar implementation work, development was begun on a Material Requisition Status System. The system's data base was logically related to the Production Control System (Polar) data base and shared many data elements. Approximately 70 percent of the data was common with data in Polar and the resulting duplication, should the data not have been shared, could have created a serious inconsistency in our systems. In addition, it was believed that by using the already established data base, system development time would be substantially reduced. This belief was justified by the fact that it required only 60 days to implement the new system. This was our initial indication, that in IMS, we had a tool that indeed offered expansionary capabilities not found in other software packages.

Engineering Release System, Phase 4 - Concurrent with our above work, we were developing the Engineering Release System, a system which provides from the engineering drawings the basic requirements for procurement of materials and the manufacture of products. The data structure for the drawing system was relatively complex. The system's forerunner had been the instrument for the development of GUAM and had provided us the technology to begin development of DL/1. The successful implementation and operation of this system did much to build our confidence in the capability and reliability of IMS.

Inspection System, Phase 5 - The Inspection System was the first new application to be implemented without its own data base. Instead of creating a

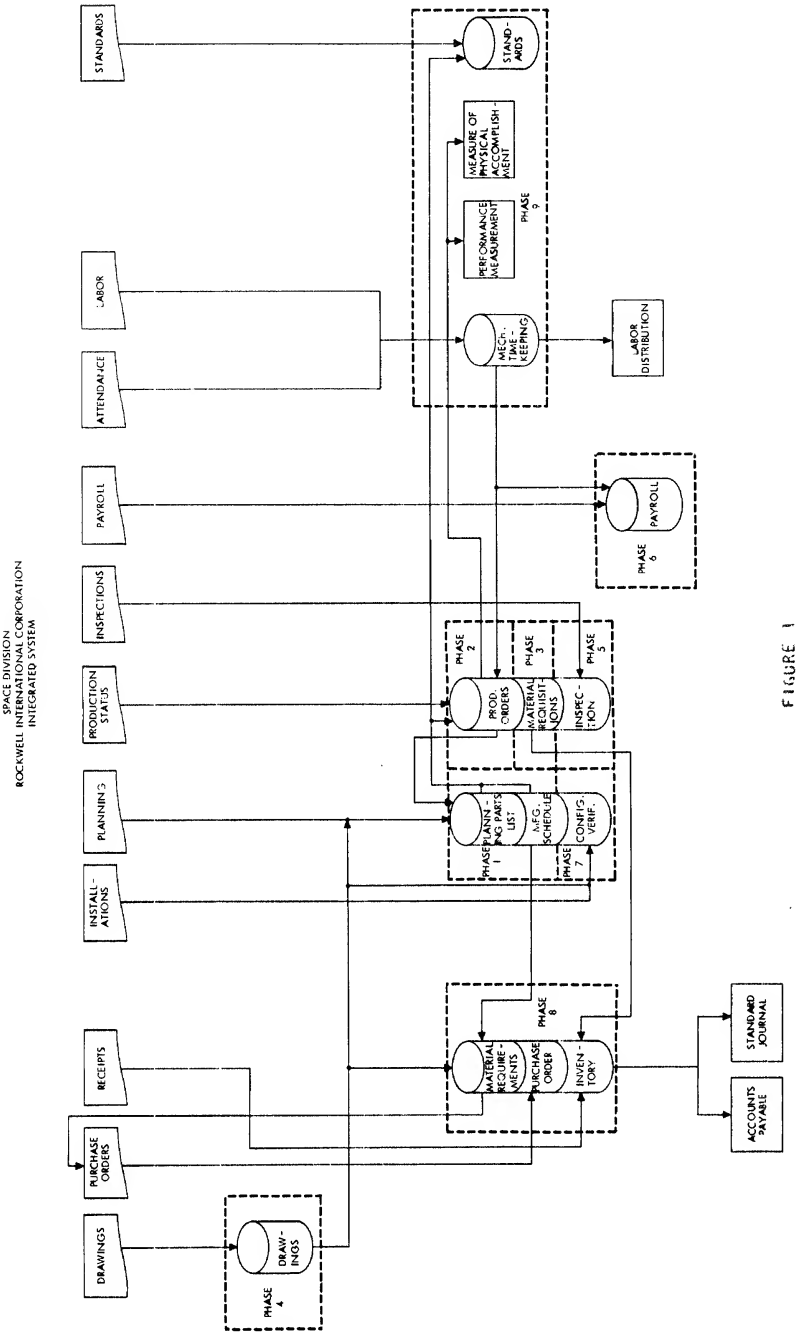

SPACE DIVISION
ROCKWELL INTERNATIONAL CORPORATION
INTEGRATED SYSTEM
STANDARDS
LABOR
ATTENDANCE
PAYROLL
INSPECTIONS
PRODUCTION STATUS
PLANNING
INSTALL-ATIONS
RECEIPTS
PURCHASE ORDERS
DRAWINGS
PHASE 4
DRAW-INGS
STAND-ARDS
MEASURE OF PHYSICAL ACCOMPLISH-MENT
PERFORMANCE MEASUREMENT
PHASE 9
MECH-TIME-KEEPING
LABOR DISTRIBUTION
PAYROLL
PHASE 6
PHASE 2
PROD ORDERS
PHASE 3
MATERIAL REQUISIT-IONS
PHASE 5
INSPEC-TION
PHASE 1
PLANN-ING PARTS LIST
MFG. SCHEDULE
PHASE 7
CONFIG. VERIF.
PHASE 8
MATERIAL REQUIRE-MENTS
PURCHASE ORDER
INVEN-TORY
STANDARD JOURNAL
ACCOUNTS PAYABLE

FIGURE 1

new set of data, a new segment was added to our production control (Polar) data base. This approach of adding a new segment to an existing data base proved extremely successful in that the programming effort required was minimal, and no redundant data was introduced into our "Integrated Information System". This inherent feature in the IMS data base management facility has been one of our key assets in systems development. It has allowed us to expand our systems without reprogramming and has made modular systems development a viable method for achieving full system integration.

Corporate Payroll/Personnel System, Phase 6 - By January, 1970, the confidence at both the Space Division and the Corporation in the capability and general reliability of IMS had risen to the point where the decision was made to develop the Corporation's Payroll/Personnel system on-line under IMS. This system was developed to handle all aspects of both Payroll and Personnel at the various divisions of Rockwell International. The decision to do this was not made lightly. Payroll data is sensitive and ability to protect and control the data was of paramount importance. No software that was not exceptionally reliable could have been considered.

Configuration Verification System, Phase 7 - By the time Phase 7 began, it had become an accepted fact that all major data base systems were to be developed using IMS. This included systems other than those planned as part of the Integrated Information System. Phase 7 in the development cycle was a configuration verification system which could compare the as-designed, as-planned and as-built configurations of our flight hardware. This system could provide an Indentured Parts List listing the production orders installed and through interface with the Engineering Release System provides us with a closed loop as-designed vs. as-built configuration.

Material Inventory System, Phase 8 - The Material Inventory Control and Information System was developed to handle material from the time requirements are identified until it is issued to the shop for incorporation into the part. The system obtains its requirements from the Engineering system and applies the Manufacturing Fabrication Schedule to determine need dates. After a buy is made, the purchase is statused until all material is received. The system is interfaced with the Polar System so that when a production order is placed in work, Material Requisitions are automatically prepared in the warehouse for issuing material.

Mechanized Timekeeping and Performance Measurement System, Phase 9 - The Mechanized Timekeeping and Performance Measurement System is the latest phase to be implemented. This system provides for our hourly employees to punch-in through a terminal rather than a timeclock when entering the plant and then provides them the ability to log-in on a particular job and record their time for labor distribution purposes throughout the day. The system then calculates actual hours worked on an operation which has had standards applied. The actual hours worked during each week are sent to the Payroll/Personnel and Labor Distribution systems. In addition, performance measurement and measure of physical accomplishment reports are prepared from the actual and standard hours.

A great deal of systems development work remains to reach our long range goals. The remaining effort is primarily in the scheduling, manpower projections, and forecasting areas. But we have indeed made substantial inroads into our long-range plan, all done in an organized manner.

CONTRIBUTION OF IMS

The development of application systems of the magnitude of these could not have been attempted without a good generalized data base management system such as IMS. Our actual requirements for such a data base management tool were defined seven years ago to fulfill our needs at that time and for several years hence. These requirments included:

1. A reliable method to access data both sequentially and direct.
2. A system that could handle large volumes of data and store it in a manner similar to real life conditions.

3. A system that could be expanded as new requirements were identified.
4. A system that would provide a maximum in application flexibility for design and programming.

IMS answered these requirements and provided the vehicle which has enabled the Space Division to progress to our present state of development.

IMS has proven itself since infancy to be a successful and viable data base management system. Let's examine the reasons Space Division has felt, over the years, that IMS has been the logical choice as a data management mechanism and is the best information handling tool in existence today. First, IMS provided multiple access methods. A single access method cannot possibly fulfill all the requirements of a complex application. A minimum of three basic access methods are required, sequential, indexed sequential, and direct.

Secondly, a data base management system cannot be a static package. In the dynamic environment in which business and its computer facilities operate, the supplier of a package must be committed to continual upgrading as the needs and requirements of the users change and grow. IBM must be complimented in this area in their earnest attempt to incorporate new user requirements, thereby providing increased capability and flexibility.

Thirdly, IMS provides a reliable data base recovery utility. Due to the increased use of data base management systems to store primary company records, a data base recovery utility is an absolute necessity. If a system goes down and data is lost either through hardware or software failure, that data must be recovered as quickly as possible with no loss of credibility. This is perhaps the major advantage IMS possesses over other packages we have examined.

Fourthly and perhaps the most significant single benefit of IMS however, is that it is a data base management system which is an integral part of a generalized telecommunication system. A data base management system cannot be developed without consideration given to its on-line use. This is where many of today's systems and/or packages fail to measure up. Suppliers have provided no telecommunication consideration during initial development. The on-line environment is much more critical than the batch mode; rapid access of data is mandatory, security problems greater, and recovery more difficult because of data synchronization problems.

IMS SHORTCOMINGS AND SOLUTIONS

IMS is not and probably never will be perfect, but it is the best system of its type. It is a large and complex piece of software; this in itself implies certain problems. the greatest of which is efficiency or speed. Unfortunately, as features and flexibility are added to IMS, overhead is increased. But IMS can be very highly tuned if some additional hardware costs are not objectionable.

At Rockwell International, we have tackled some of the major areas we felt were unnecessarily slow and provided new utilities and modifications to increase efficiency. One example of this was that the sequential processing of an on-line data base for either the producing of reports or a sequential dump was extremely slow. An examination of the code indicated that most of the code being executed was for generalized conditions which seldom occur. A utility was written which eliminated the general code. The result was that we were able to sequentially process a data base in one-fourth the time. This utility now logically processes a data base only two percent slower than a physical dump.

IMS does not have every feature that we would like to see provided in a package of this type (nor does any other, for that matter), such as Inverted Lists, on-line system performance measurement, etc. In some cases we have developed features for our own use; in other cases the additional feature was determined undesirable due to additional system overhead. In a few instances, our requirements were made known to IBM for their action and were eventually statisfied.

Some of the enhancements that we have developed at Space Division include two fast dump utilities mentioned earlier, one provided with exits for the

applications programmer and the other without; an on-line system performance measurement routine; a message interceptor to permit the applications programmer to analyze this message and choose a particular data base for updating; a data base dump routine to selectively print DL/I data bases in a readable format; and a generalized report generator/query facility for IMS data bases.

This latter capability is one of the principal IMS enhancements that we have instituted recently. This major software uses a list directed modular approach for generating the code to produce multiple reports from IMS or non IMS data bases. Some of the advantages this software has over other available tools are its ability to generate reports from multiple hierarchical paths, and from multiple data bases either on-line or off-line in professional type format prepared completely within the software. In addition to the report generation facility, the system, using inverted files, can answer user on-line queries about his data base without the need for searching the entire file.

Although IMS failures occur infrequently, as in all systems, they do occur. The reliability factor in a production environment, such as at Space Division, is highly critical. If the system has a major failure it is possible for the Division's work to come to a standstill. Because of our concern for absolute data integrity, we have had to skip implementation of certain IMS releases because of indications from our tests that they were not as reliable as our environment demands.

What we feel are shortcomings of IMS though are not unique to IMS itself. Realistically, we all know that every data base management system available has some problems with either speed, reliability or feature availability. We believe that IMS is the best system of its type. Our extensive experience with IMS has shown us that it offers adequate data base access coupled with an outstanding telecommunications package that allows for system development ease in the on-line mode, a requirement in today's fast paced business environment.

IMPACT OF IMS

A significant change has occurred in systems development and operation at Space Division as a result of IMS. Greater time is being spent in system and data base design than previously, because system performance using IMS is not determined by the skill of the programmer but rather the structure and access mechanism for the data base. Commensurate with this, the skill level of programmers can be lower, the time required to program can be greatly reduced, and the overall result is a reduction in system development costs. An interesting element of using IMS is that a much more critical evaluation must be made of the systems analysis capability in one's organization. The longer we are involved with IMS, the more obvious is the gap between the talented senior analyst and the mediocre designer. Indeed, it is critical to apply the organization's top analysts to the IMS task.

Another interesting aspect of our IMS environment has to do with system operational costs. Since implementation of our first IMS on-line application numerous production cost studies have been made to compare on-line and batch operational costs. The most extensive of these studies was made at the time we converted the Manufacturing Planning System to on-line IMS. At this time, we had an opportunity to directly compare on-line and batch costs for the same system. The results of this study indicated that the machine costs for processing were nearly identical, but the manual costs were substantially lower for the on-line system. The net result was that the on-line IMS system costs were only one-third of the batch system costs, a statistic that deserves major consideration.

Rockwell International has had a great deal of success in developing applications using IMS as its data base management system. It is our opinion that IMS answers a real and current requirement for a data base management system and with the continual enhancement of the system by IBM it appears that it will be a satisfactory system for many years to come.

DISCUSSION

TURNER:

Some people think that IMS is a very large system and that it requires a very substantial amount of systems programming support to bring it up and to keep it running. Would you care to comment on that?

LIGHTFOOT:

It is a very large and very complex system. We're supporting the system as you see it plus about four other divisions' worth of work with two systems programmers, one handling communication and one handling data base.

TURNER:

Many of the data base management systems have an integrated data dictionary facility. Do you feel that might be a desirable capability?

LIGHTFOOT:

I think it is a desirable feature. However, I look at the requirements for maintaining it and I question if you are not getting into a real maintenance type problem that would not be better done separately.

TURNER:

How do you feel about the facilities for data definition? Particularly do you feel the current mechanisms within IMS for describing data are what you would call desirable?

LIGHTFOOT:

Even in our environment we do not use the full capability that IMS has for defining data and I'm not sure that it's not a massive documentation problem. I'm evading some of these questions because even with the literally thousands of data elements we have had, we have been successful in documenting them as individual modules rather than as a separate package, and it seems to have worked for us. I would say that it does not document the data elements in the structure as well as it possibly could.

REINSTEIN:

I want to ask you how much data sharing, if any, there was between your subsystems, and more generally do you feel that there needs to be a greater amount of data sharing function in a data base management system.

LIGHTFOOT:

We have done it in different ways. The production control systems, which were Phase 2 and Phase 3, were done in such a way that they actually shared the data between the data bases. Our material status system essentially statuses requisitions and supporting production orders so about half of its data is identical to that data that is in the production data base. Indeed when we designed that system we designed two separate data bases with the same key and shared the data in the different applications, but we just performed two accesses to get the data. This worked exceptionally well. On the inspection systems we just added another segment into the data base to come up with a complete inspection record for production orders. I would recommend strongly against putting too much data into a single data base. The cost of direct access is so low and the reliability and performance improvements you get by duplicating data

is so great that I think it is advantageous to duplicate the data. I was working with a customer overseas who had read the IMS Manuals on duplicates and redundant data and had come up with some designs of data bases. They literally followed the text book. No redundant data in the keys, a perfect type structure right from the IBM Manuals. We put in some redundancy in keys and redundancies in data on the single data base, doubled the size of the data base and cut the message region CPU time in half. We happen to feel processing time is more expensive, therefore we get a benefit.

WORLEY:

In answering an earlier question you cited performance as a reason for preferring duplication rather than increased sharing. Are there other specific things that you observed that would lead you in the direction of duplication?

LIGHTFOOT:

The recovery situation is very critical. The way our data base and systems were designed, the duplication of data allows applications to continue to function even though one data base may be damaged. You may require data in two places and they may be completely different as far as organization. Engineering may require part-number information as well as manufacturing. If you share part-number in the two worlds and the data base goes down, you shut down manufacturing and engineering. In any complex environment you will have failures from time to time, so the duplication of data tends to give you a hedge against failures. It brings down less of your system by duplicating some data. In addition the duplication of data also allows you to recover a little faster in that you have a smaller data structure.

JARDINE:

The problem you are trying to solve is that of performance and reliability. Duplicating data to solve these two problems is not the right way to go about it and I'm sure everybody in this room understands that. Instead of duplicating data, what we had better do is really solve the two problems.

LIGHTFOOT:

Don't forget to throw in the third problem of flexibility.

JARDINE:

I'm not certain that you lose a lot of flexibility. If the data is there and is available at a reasonable cost then I don't understand the flexibility issue at all.

LIGHTFOOT:

I think generally everybody agrees with you. However, the point I have is that the applications vary in their requirements for data retrieval. Some applications require only direct access, others only sequential and others some variations between. No company to date has developed a single access method which has what I consider satisfactory performance for both sequential and direct access. I think we would be a little naive to believe that such an access method is imminent. Millions of dollars are tied up in systems which have been developed for a specific type of access method. A change in the access method could cause a major reprogramming effort.

Data Base Management Systems, D. A. Jardine (ed.).

USER EXPERIENCE - "TOTAL"

W. E. MERCER
Eli Lilly and Company
Indianapolis, Indiana

INTRODUCTION

Each person who is a "string-saver" or who is married to a "string-saver" should be required by fire regulations or by good sense to move every 3-4 years. For the act of moving forces not only the discarding of some obviously unusable articles but mandates a review of why certain things were saved. This review often brings on a certain level of nostalgia and a recall of days of youthful innocence. I use this analogy in presenting a paper on user experience, for the act of preparing a paper again forces that review, an attic cleaning time of ideas perhaps.

This paper will discuss why we installed a Data Base Management System, the selection of TOTAL, what is TOTAL, our mode of operation, provide operational statistics and draw some conclusion as to what we have learned from installing a Data Base Management System.

ABOUT THE COMPANY - ELI LILLY AND COMPANY

A manufacturer of biologicals, antibiotics and pharmaceuticals in human health products; empty capsules, industrial chemicals, animal health products and agricultural chemcials in our Elanco Division; cosmetics in our Elizabeth Arden Company; and plastic boxes and retail packaging cartons in our Creative Packaging Company. We have worldwide operations both in sales and manufacturing; however, we originated in and have tended to retain the majority of our manufacturing facilities and corporate headquarters in the central Indiana area.

The data processing issues for Data Base are those of any large manufacturing company: Bill of Materials, production scheduling and shop floor tracking, inventory control, Financial Accounting, Market Analysis and Reporting, Requirements Planning, and Personnel Data. Perhaps we are a bit different than the layman's concept of a manufacturer, in that our Engineering Division is Chemical Research with its own very unique problems including many reporting requirements to the Federal Government.

WHY WE INSTALLED A DATA MANAGEMENT SYSTEM

Reviewing our original justification papers to management, as well as our working papers as we explored the idea of Data Base Management, was a nostalgic visit to yesteryear. How well we have accomplished these objectives I will leave for the summation at the end of the paper. Some of these judgments of accomplishment will be left to your intuitive reasoning for I come to praise Caesar, not to bury him.

We believe we should install a Data Base Management System for the following reasons:

1. We would be able to develop a standard approach for systems design and communication with corporate data bases.
2. We would eliminate redundant data that existed in our (then) present files and the attendant duplication of file maintenance.
3. This would give us the capability so that in the future we could design

integrated systems.
4. By virtue of the fact others were concerning themselves with data description and control we would increase the productivity of the application programmers.
5. The data independence feature of Data Base Management Systems provides for greater efficiency in the revision and maintenance of production programs.
6. We could provide a general knowledge as to location and content of available data for timelier management reporting.
7. A standardized approach would be available for data base backup and recovery, also we would be better able to insure file integrity.
8. A centralized data base and data base management system would better allow centralization of processing at the corporate computer center.

SELECTION OF "TOTAL"

Now once you have committed yourself to the unassailable rationale as to how outstanding you are going to make the world if you could only install a Data Base Management System, by golly you had better find one -- and a good one at that.

After a rather lengthy review and information gathering process, an in-depth reporting of that study perhaps would be worthy of a separate paper, TOTAL was selected for a two application trial. It is realized that application oriented files are a contradiction in terms to the universal approach implicit in Data Base Management. We, however, wished to adopt a low risk posture. The trial began about February 1971. The significant feature of each of these application area was though extremely diverse in their orientation, they both had long due dates. Thus, it was felt that we could backup and regroup if TOTAL, despite our evaluation, proved to be not a viable Data Base Management System. The features of TOTAL that influenced its selection were:

1. Modest allocation of resources required at the entry level. I believe our initial TOTAL region was in the 40-44K region size. At that time we had an MP65 and each core box of 256K rented for around $10,000 per month.
2. TOTAL interfaced with CICS. We had been using IBM's CICS as a task monitor system for our teleprocessing metwork for over a year previous; thus, had a rather major investment in programming by this time.
3. The throughput ability of TOTAL was governed by state of the hardware art. Its software approach to Data Base Management recognized the rather primitive state of hardware in this area and attempted to maximize throughput. In the design of its indexing techniques throughput was a major consideration and no exotic indexing was attempted if it is not a good trade off with throughput.
4. It provided data independence to the application programmer a prime requirement.
5. The approach to data relatibility was one of networking. This was importantly compatible with our conceptual understanding of our data base structure. While hierarchical association would probably have been workable, it appeared foreign to our thought process on relationships.
6. It was quite easy for our application programmers to use. It could provide data to an application written in any language capable of a call function. On the IBM 360/370 these are BAL, COBOL, PL/I and FORTRAN.
7. TOTAL provided data integrity for concurrent on-line (CICS) and batch processing against the data base.

WHAT IS "TOTAL"

It seems a bit out of order to discuss our decision process for the selection of a Data Base Management System without providing my audience additional insight into precisely what it is and how it operates.

This reminds me of a sweet young thing, sweet is feminine in this story, standing beside two erudite gentlemen at a cocktail party who were discussing

Keats vs. Shelly. Finally she could contain herself no longer and asked one gentleman -- What are Keats?

TOTAL is a Data Base Management System that will manage virtually an unlimited number of data sets. The word data set is an important issue here for it does rely on its knowledge of physical contiguity from the data descriptor code to construct a logical record. Also, the application programmers must know which data set the data of interest is in, for the programmer must provide the data set name in their call for data.

Association of data is again at the data set level. Any data set can be related to any other data set. There are some deviations from this - coded records - where a data set can relate to only certain records of another data set. This is a very powerful and throughput oriented feature of TOTAL and not to be passed over lightly. However, this is not intended to be a tutorial and those with further interest may explore it on their own.

This association of data provides for network or associative structure. I presume it could be thought of in the parent-child relationship so traditional in hierarchical structures. This would be a bit contrived or artificial; however, if it were beneficial to your logic, I can't believe it would be harmful.

TOTAL provides data independence below the data set level. In response to a data call, a logical record of only those elements of interest is provided the problem program. No information concerning data set linkage is inserted in the problem program.

TOTAL provides for evolutionary development of data base since additional data elements or data association linkages can be added without impacting existing programs. To add additional data elements or create new linkages the file must be dumped and reloaded in an expanded disk area.

MODE OF OPERATION

TOTAL occupies its own region. All data sets managed by TOTAL are assigned to that region by the standard 360/370 OS conventions of DD cards with standard JCL. By occupying its own region, TOTAL is operating under its own storage protect key and is insulated from the problem program.

Tasks requiring the services of TOTAL sign on to TOTAL's Task Table by means of an initial call. After sign on TOTAL constructs a logical record in the problem program region in response to a request for I/O. The content and physical order of the data within the logical record is a function of the problem program.

Various facts concerning the physical record, hardware device it is stored on, data elements, and association linkages are described and compiled separately. They are loaded by TOTAL into its region from a partioned data set.

OPERATIONAL STATISTICS

We have 11 application areas using 116 files under TOTAL. These applications run the gamut of areas of interest you would expect from a manufacturing company. General Ledger Accounting, Product Distribution, to a relationship of items to required assays. Some of these files are common use, however, at this time many are specific to an area of interest.

Of these 116 files, 74 are single entry files or files with only one entry point and one record per entity identifier; 42 are variable entry. These can have multiple records per entry and can be related to many single entry files. The 30 single entry files without associated variable entry files tend to be information files. For example, it could be a State file from which, by keying on State code, you could extract various information about a State from the name to the sales tax.

From a recent 30-hour running period on a 370 Model 155 we have extracted the following statistics.

- 1,800 elapsed minutes
- 206 cpu minutes used by TOTAL
- 1,280,000 physical disk accesses
- 6,021 physical access per cpu execution minute

The physical accesses to TOTAL files are measured, the logical requests made to TOTAL for data are not measured. However, the expected value of such requests based on our sample is at least double the physical accesses to the files.

Of the 11 application areas using TOTAL, 9 are on-line under CICS. All of the 11, including the on-line systems, have batch processing requirements. It is an interesting phenomenon that even though an application begins life as an on-line system it will in the end generate batch processing report requirements. Evidently a quick peek in a CRT is no substitute for a voluminous formated report.

TOTAL at present is in a 132K region. Approximately 8K of that region is used for OS Control Blocks, 22K for the Executable Code, 48K for the data descriptor module and 52K for I/O Buffers. All data is on 3330 disk units at quarter track blocking which is approximately a 3K block.

WHAT WE HAVE LEARNED ABOUT DATA BASE MANAGEMENT

A prime requisite for success in most ventures is a competent staff; Data Base Management is no exception. We have identified the need for both technically and systems oriented people. If you can acquire a staff of people with these orientations in tandem, you will have a competitive edge.

Looking first at the technical job functions, for those are the easiest to discretely identify, these are: utility programs, care and feeding of the Data Base Management System, systems performance measurement, disk allocation and mapping, and applications programmer consultation.

From your various vendors you will acquire a certain level of utility programs. However, each installation will generate additional requirements for programs to move, validate, count and re-arrange data. Care and feeding of the Data Base Management System consists of problem definition, interfacing with the vendor in problem resolution, and installing new releases.

We have adopted a slogan that goes: "Show me a System 370 with poor throughput and I will show you a machine with poor disk allocation." To maintain your system at something near optimum performance level, staff effort to measure systems performance and attend to disk and data set allocation is required. Various hardware and software monitors are available to aid in measurement. Don't dismiss lightly the requirement to be available to the application programmer in a technical consultant role. The programmer, who added John to Smith instead of 1 to 100 and got a data exception, suddenly falls apart if the record was presented by a Data Base Manager instead of a reel of tape.

As mentioned previously, systems oriented jobs for the Data Base staff are not as readily defined as the technical. Their assignments are in the area of file development. Interfacing with both the end user and the applications staff they can be more readily identified by skills required than assignment. These are: communication skills, political sensitivity and good at coding techniques.

We will hear and have heard comments on the requirements for data independence. Today's Data Base Management Systems have not yet achieved independence to a proper state of user transparency. Certain data knowledge continues to be buried in the application program. Due to this, correction of a data base design error cannot be made transparent to the application programmer. Thus, the application programmers and/or their management must participate and agree to the reprogramming effort required to recover from certain design errors. In general, the reward system for application people is not based on reprogramming operational systems, so potential conflict exists in this situation.

Additional brief for data independence can be presented that outranks the transgression of simple design errors. That is, Data Base design is an evolutionary process. Establishment of data files cannot wait until all possible facts about the data are determined. Consistency or standardization of coding

practices will aid in both separating and combining presently existing data base files where we simply did not understand the problem or the software or hardware technology was not adequate for current needs. However, changing real world relationships are unavoidable in the structure of the data base considering the dynamics of our current environment.

SUMMARY

We have a Data Base Management System which is current state of the art. We have learned much about staffing requirements, interfacing with the user and application programmer community. Foremost, however, we have learned that additional data independence is necessary to truly advance in the art of Data Base.

DISCUSSION

MULLANEY:

In the conversion to TOTAL one of the things that concerns us is the loss of efficient retrieval of sequential datasets. How much of an effect was this?

MERCER:

A TOTAL file is a standard O.S. dataset. You can access it outside TOTAL. If you access it through TOTAL there is a certain level of efficiency lost, there is no getting around it. I can't tell you what the efficiency loss is however.

STEEL:

From your description of what has to be done to add a data element, it seems that TOTAL is effectively unusable for very large files unless the files are going to be fixed forever.

MERCER:

We can dump and put back pretty fast, but if you're talking on the magnitude of Equitable, it would be a consideration.

STEEL:

How long does it take you to dump and restore a file? (Pick some numbers.)

MERCER:

I would say between 2 to 3 hours at times to rebuild one and I suspect it has been beyond that.

STEEL:

How big a file would that be?

MERCER:

You ought to be able to load about 60,000 records an hour. There are some techniques you can use to get up to about 240,000 an hour.

KURZ:

Would you comment some more on redundant data.

MERCER:

We are going to do away with it. We have not done as well as we thought we would. I'm going to be very pragmatic on this. To lose a file is a problem, but if you only shut one or two people off, you can live with that. If you shut the world down when you lose a file, you're in a lot of trouble. I know some people who have had this happen. I like to keep a low risk posture and I think that would be a high risk since redundant data does aid in file reconstruction.

SCOTT:

Would you like to point out any particular shortcomings?

MERCER:

The knowledge of the particular linkage the programmer has to chase is embedded in his program. This is a problem for us. We've made a design error. looking ahead six months and knowing what we're trying to do, I can see that we've got a major design problem. The linkage is going to change and I'm stuck with programs with this linkage path name.

SCOTT:

I presume you have the need to process concurrently with several programs. Is the support for this adequate?

MERCER:

Yes, no problem at all. It will inhibit and protect you from concurrent updates. Doing this might drive your teleprocessing users up the wall. When they keep looking at the data and find it changing as fast as they look, it takes them a while to realize what's happended to them. In general we have prohibited this because we don't want to drive people crazy, not because there is anything wrong with it.

SCOTT:

Given a chain of employees do you have the ability to directly access any employee in that chain without having to search the entire chain on some alphabetical order?

MERCER:

Yes, you can go right to that employee number, even on variable files, where we have multiple records for employees, say one on salary, one on education and so on. Given employee number you would get all possible records for that employee.

KAMERMAN:

The real test of data base design is not the first application using a particular data base, but how well you can put new ones on that base without causing a major reprogramming problem, or redesign problem.

MERCER:

We have not had that yet, with the exception of one customer file I mentioned.

KAMERMAN:

You said that you had a problem with its structure or design. What about the utility of an integrated dictionary in that kind of environment which would perhaps get you out of having the data dependencies, the linkages, or whatever within the program?

MERCER:

I think we need one step more insulation than we have now and whether that's in the dictionary, whether it's in the interface design, I'm not certain. I certainly need more insulation than I currently have.

GRAVES:

You promised us a summary of how well you did in your objectives.

MERCER:

Standardize systems design, no. Eliminate redundancy: we decided that wasn't quite as great as we thought it was to begin with. Future capability: yes, we have built a lot of systems on existing data and continue moving forward in that direction. Increased productivity: yes, we have increased the productivity of the application people in those areas where we are using TOTAL. There's a lot of things application people do besides access files, so we haven't been attending to that. Program maintenance and revision: yes, they've added fields and so on and we haven't had to go back to the drawing board and recompile all those programs. Timelier management reporting: I doubt it, however, I believe this is ripe fruit that will be picked. Standardized backup and recovery: yes we've done a good job on that one. Centralization of data processing: yes. All of our remote sites are losing their local capability. Over a period of time we will have no processing at remote sites except R.J.E.

Data Base Management Systems, D. A. Jardine (ed.).

USER EXPERIENCE WITH INTEGRATED DATA STORE (IMS)

G. L. VON GOHREN
The Pillsbury Company
Minneapolis, Minnesota

I am speaking representing both the Pillsbury Company, which is a heavy user of IDS, and as chairman of the Honeywell Large Systems Users Association Database Committee. In that capacity I have become familiar with the progress of many other users.

SELF-INTRODUCTION

I wrote my first computer program without the benefit of an assembler on an IBM 650. That was as a student at the University of Washington in 1960. My next association with computers was as an employee of the Weyerhaeuser Company which was developing an M.I.S. on General Electric 225's and an I.D.S. data base. This was late 1963 and all coding was done in assembly language. In 1966 Weyerhaeuser began a joint development with G.E. to expand the GCOS operating system on multiple G.E. 635's. The system was called WEYCOS and I did much of the IDS modification.

I went to Pillsbury in 1969 primarily for the development of a remote Query language for IDS. In 1970 I became chairman of the User's Data Base Committee. My current activities are in system efficiency analysis and making data available to users from the data base via Computer Output to Microfilm.

PILLSBURY PROFILE

Pillsbury has 18,000 employees, net sales of $816,000,000 and is headquartered in Minneapolis. The company is divided into five major businesses:

Consumer	Family flour, packaged mixes and refrigerated dough products.
Burger King	850 plus, fast food locations.
Agri Products	Flour milling, Industrial sales, and commodity trading.
International	has 23 operations in 15 countries and exports to 90 countries
Poultry	fresh chicken and feeds for growers

In addition, there are smaller, growing businesses in fresh flowers, specialty restaurants and production and import of wine.

The corporate computer facility services all except Burger King, which has its own system, and International. Current hardware is a dual processor H-6080, 256K words of memory, 150 remote terminals, 1 billion, 19 million char of disc, 593 million of which is IDS, and a store and forward switching computer.

PILLSBURY JOB LOAD DATA, MAY 1973

Total Activities	42456
Non Billable Activities	384
TSS Hours	2316
TSS Connects	11770
Total Application Executions	22632

Total IDS Compiles	665	
Total Cobol Compiles	391	
Total File Dumps	1774	* files are segmented for dumping.
Total File Loads	662	* includes weekly system reload and some testing.

OTHER TYPES OF H-6000 USERS

University - student and administrative
Auto manufacture - parts resupply
General manufacturing - wide range
Brokerage - inquiry and transfer
Banking - trust and demand deposit
Military - strategic and financial
State govt - resources data base, auto registration, law enforcement.

IDS INTRODUCTION

IDS was developed on the GE225 in 1962. All procedure coding and data definitions were in assembly language. In 1966 a COBOL imbedded language was developed for the GE400 and the new 600 line. It used a prepass translator to convert the IDS verbs to assembly language calls and build an assembly language structure for the data definitions. These are passed directly through the COBOL compile with suitable equates to COBOL references and working storage.

A typical compilation looks like figure #1. File data description may be supplied with the source, or drawn from a library. There is no copy equivalent to COBOL copy so users have developed several systems to do this. A typical execution core map looks like figure #2.

Each buffer holds a page of IDS records. Each page is logically numbered and constitutes the physical unit of disc transfer. Although pages may be created in various sizes, a typical one is 1920 characters and can hold up to 63 records or line numbers except as limited by space. The combination of page number plus the line numbers as a suffix constitues the reference code. Reference codes are used as the pointers of links between records to form chains.

IDS uses a shorthand method to describe hierarchical and network relationships between different types of records. Each record must be either known by its reference code (primary record) or as a member of one or more chains each of which emanates from an occurrence of a master (upper hierarchical) record and includes none, one, or more details. (Yes, even randomized (CALC) records belong to a chain. They are randomized to a page and then linked in a chain emanating from that page header allowing overflow into other pages if necessary by just continuing the chain.)

The next few figures show structures, first in "longhand", then in "Shorthand". You can soon see the need for abbreviation.

This example shows in shorthand a moderately complex data base that might be used by Pete Rozell in managing the business of the National Football League. It is a data network; far more complex than a sequential, random, or tree structured hierarchy. It also includes an example of each of the basic structure relationships.

IDS verbs themselves are quite simple--retrieve, next, prior, each, current, master, random, store, modify, delete, move (to working storage). All housekeeping and much implied activity is triggered by the verbs.

ENHANCEMENT HISTORY

IDS has remained much the same in terms of data structure and procedures since 1963. The conversion from assembly language on the 225 to COBOL as a host

language was the "second generation". Enhancements have been primarily "add-on" and efficiency improvement. Major examples are:

Pagettes can be used for file expansion when pages allotted to a particular subschema become filled with relatively few large records leaving many of the 63 line numbers in a page unused due to lack of space. By subdividing line numbers into subsets a page of 63 lines can be physically extended over several pagettes. This helps solve problems due to unexpected growth which can be a real burden if not initially allowed for.

Randomizing Analyser - This utility can perform three important functions:

1. Determine if random records to be loaded will skew or randomize unevenly in a given file range.
2. Pre-sort random records to be initially loaded in a file into reference code sequence--resulting in much reduced I/O and elapsed time.
3. Pre-sort heavy volumes of randomly processed transaction into reference code sequence to reduce I/O.

Data Base Utilization Analyser - After scanning either the data base or its dump tape, this utility reports population of each record type, % space fill, and % line number fill.

Execution Statistics Monitor - Reports the number of executions, number of resulting I/O's, and record types and chains processed for each IDS verb after program execution. This may provide clues to unfavorable file conditions such as overcrowding, inappropriate use of a given file structure, too few or too many program buffers, or improperly sequenced program logic.

Data Query - IDS data query is a self contained language developed jointly by Honeywell and Pillsbury. It provides a simple language with optional tutorial output to retrieve and display a few instances, build a file of many virtual records meeting retrieval criteria for further processing, or resequencing and formatting a printed report. Data query at Pillsbury will run under time sharing for short retrievals or spawn a batch job if expected to be long. It is used by field and office non-data processing personnel.

Data Query represents a step towards file-contained, as opposed to program-contained data structure definitions. Based on the datanames references in the inquiry, it determines what subschema is implied, retrieves the partly assembled structure components from a library and pastes together a suitable structure definition for a specific execution. The original version did not have the ability to "head" a chain (to retrieve a master of another chain which is also related to the current detail record). That capability is now being implemented at Pillsbury.

Multi-Access-IDS - MAIDS allows multiple programs to update a file concurrently. The system is oriented around a Clean-point and an Inhibit/Enable declaration by each updating program. A clean point usually occurs between transactions, or groups of transactions when a program could logically go to end of job. At that point, buffers are flushed back to the disc and a journal entry is made. This is an aid to establishing recovery points. Inhibit/Enable verbs control the access by other programs while the inhibiting program performs a sensitive operation.

Programs may concurrently update without using the MAIDS verbs. However, the automatic protection is only to the extent of providing the most current record image. Manual analysis must be done in order to keep another program from becoming dependent on the first program's successful completion or operating while some invalid condition exists due to another program's activity.

This implementation by Honeywell, as well as projects by several users, indicates that multiple update runs best in a transaction program environment. One long-running batch program might well be run concurrently with other transaction type programs as long as it made frequent MAIDS verb declarations.

Transaction Processing System - TPS was developed because there was no good remote IDS access method for the user whose computer was "all things to all people". COBOL IDS programs were typically too large to run compatibly in the time sharing environment and time sharing didn't have adequate interface to

journalization and batch IDS file access controls. Direct Access, the capability of a traditional batch program to talk directly to a terminal required too long to get the program allocated unless traffic for that application was heavy enough to justify leaving the program core resident. Were I to list the features of TPS, none would pertain directly to IDS. The net effect is to provide a better environment in which to operate a wide variety of remote transaction applications.

Index Sequential Processor - ISP was made available almost two years ago. It has become a good alternative for sites which hesitated to use IDS for a large but simple file and were not committed to a total data base concept. A distinguishing feature of this implementation is the internal use of pages and pointers to establish data records logically at a certain location. Combined with a specified % fill declaration for each page at creation, any overflow searches are broken into small search segments.

Databasic - Databasic was developed by Honeywell and has only been available on a test basis. It uses IDS internally to operate on an inverted data base from a time sharing BASDIC-like language. Quite complete small systems can rapidly be built with it. It provides a fast way to model an application (though not the file structure) during a feasibility study.

My impression is that it will never become a fully supported product. The users are afraid it's going to be dropped and therefore don't use it. Honeywell says, "Why support it, nobody's using it?".

User Software Developments - WEYCOS - Twenty five per cent of the effort on this joint Weyerhaeuser G. E. effort which began in 1966 was concerned with IDS. Features added were:

- File size increased from 2^{18} to 2^{24} pages.
- Automatic rollback of data bases on system or job abort.
- Multiple update with dependency prevention.
- Test updating of production files using auxiliary pages and cross reference to changed pages.

This transaction based, communications network system is still operating, but since the rest of the vendor software keeps changing, WEYCOS is slowly losing favor. Over time, its features are gradually becoming available in the standard Honeywell product.

Simplified OnLine IDS - SOIDS was developed at General Electric Ordinance Systems during 1970. Its objective were to:

- Provide on-line IDS inquiry and updates.
- Overcome Direct Access objection mentioned earlier by operating object programs as time sharing subsystems.
- Utilizes a simple command language for non-programmers.
- Produce small runtime object programs to avoid core contention with other time sharing programs.
- Operate on simple sequential files in addition to IDS.

SOIDS has been widely used by many G. E. installations and elsewhere.

ZOURCE - Zource is an IDS based system to hold compressed COBOL source decks, IDS data definitions and other code to be copied into a compile input stream, and object decks. It was built in 1969 by a Honeywell field support employee to get away from card handling and to supply the much needed IDS copy capability.

Many users are currently using this system. Recently, Pillsbury has been using the time sharing editor to maintain its source files. Zource is used only for IDS copy functions.

Alcoa - Operational since 1970 at Newberg, Indiana is a multiple update, transaction executive with automatic data base recovery and transaction reprocessing. This system was a forerunner of both the Honeywell Transaction Processing System and MAIDS. The executive is similar to time sharing in that it manages a swap area and does the remote I/O. In addition, it journalizes both the data base "befores" and the transactions. It allows one batch job to run concurrently with the transactions and is dependent upon "cleanpoint" declarations by the batch job.

Restart and Recovery - Most users are of the dump, reload and reprocess school. They seem to like the simplicity and visibility of it. Journalization of Befores

and Afters is available but used by few. At the opposite end of the scale are Alcoa, Weycos, and those users that journalize on disc or tape for each job instead of a central system journal. They sometimes include abort logic in their job control language to automatically execute optional reloads of their file.

USER QUOTES, PHILOSOPHIES AND OTHER COMMENTS

One user said, "It takes a new user about four years with IDS to become aware of its shortcomings". To further explain; by then he has a large data base and many programs using it. There are a number of changes he'd like to make but the lack of a restructuring utility and general high cost of modifying programs is a deterrent. A change in IDS logic to correct a long-standing minor internal problem is proposed. The user has to turn it down because he hasn't had enough control over past programming to know if someone has somehow coded their way around the problem and would now be adversely affected by the new changes. The user determined that since IDS used COBOL as a host language, all COBOL programmers would become proficient in it. He is now reverting to a much smaller group of people doing most of the COBOL and all of the new IDS applications. He wasn't able to train, set standards and effectively control that large a group of programmers.

The underlying reason for much of this is the power and flexibility of IDS. Since IDS is implemented as an in-line, procedural language, a quite complicated procedure can lie hidden in the many pages of program listings. Also, a great deal of implied processing may result from the unwary coder or designer's actions.

Pillsbury's emerging direction is to implement a data base structure, data capture, and update systems and then give the user retrieval tools to do his own exception reporting and inquiry. This also appears to be a trend to larger data bases, since all reasonable data is included in its most natural relationship instead of including only that which is needed by currently defined outputs or reports.

CONCLUSION

Improvements for many of the shortcomings I have mentioned may be just around the corner. I suspect that the eventual appearance of a Data Base Task Group oriented product will both provide solutions and lag our technical horizons so that we are again tempted to complain. Certainly it will bring about a file restructuring facility.

Another question to be asked if and when an IDS replacement appears is; what about conversion? The following table gives a rough idea of the potential differences:

IDS/DBTG LANGUAGE COMPARISON
(BASED ON CURRENT CODASYL DOCUMENTS)

DATA DESCRIPTION

Equivalent functionality - 88% of statements
Equivalent structure - 80% of statements
Non Equivalences:
- overlapping and dynamic page ranges
- record physical placement override
- initial interval between records

DATA MANIPULATION

Equivalent functionality - 83% of statements
Equivalent structure - 76% of statements

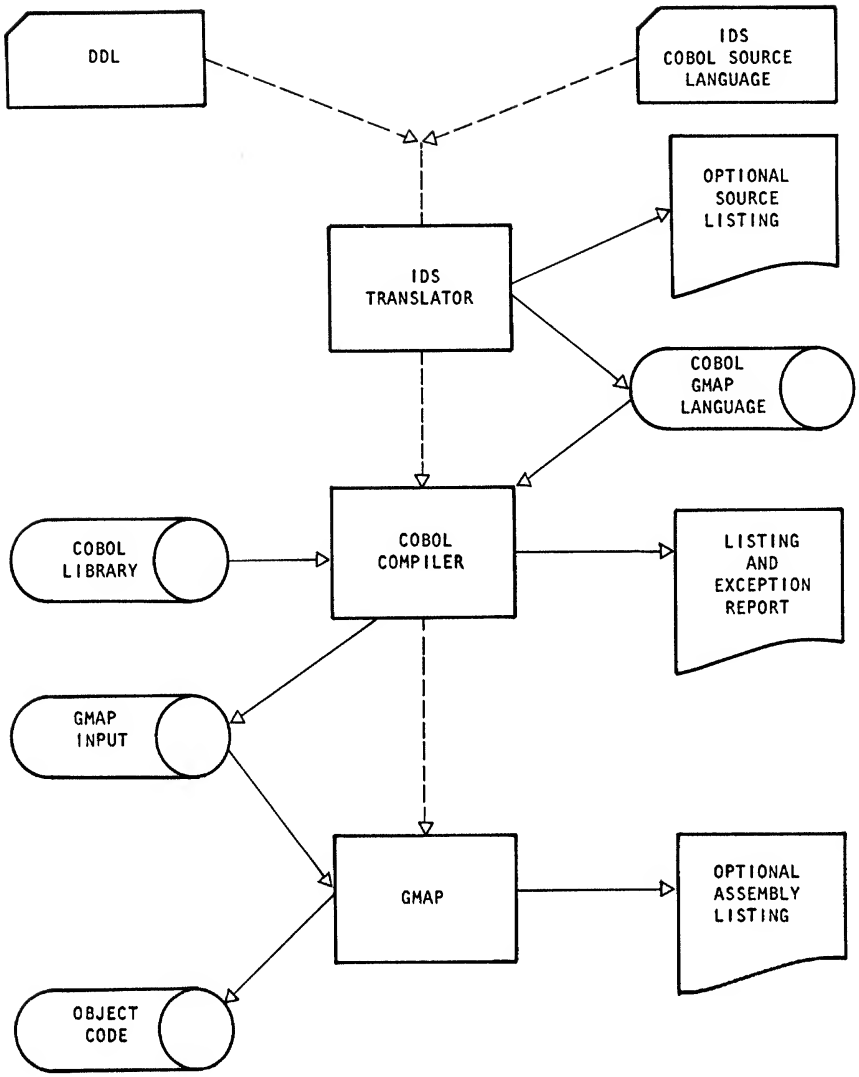


FIGURE 1 I-D-S COMPILATION PROCESS

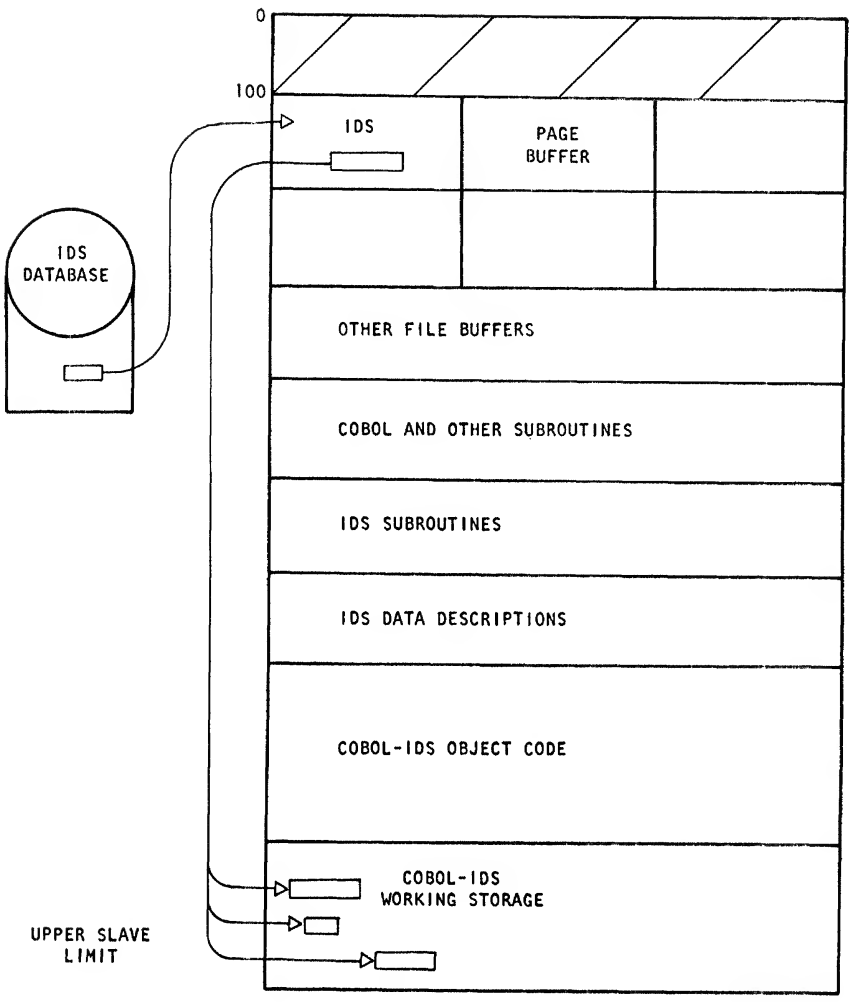


FIGURE 2 TYPICAL IDS EXECUTION CORE MAP

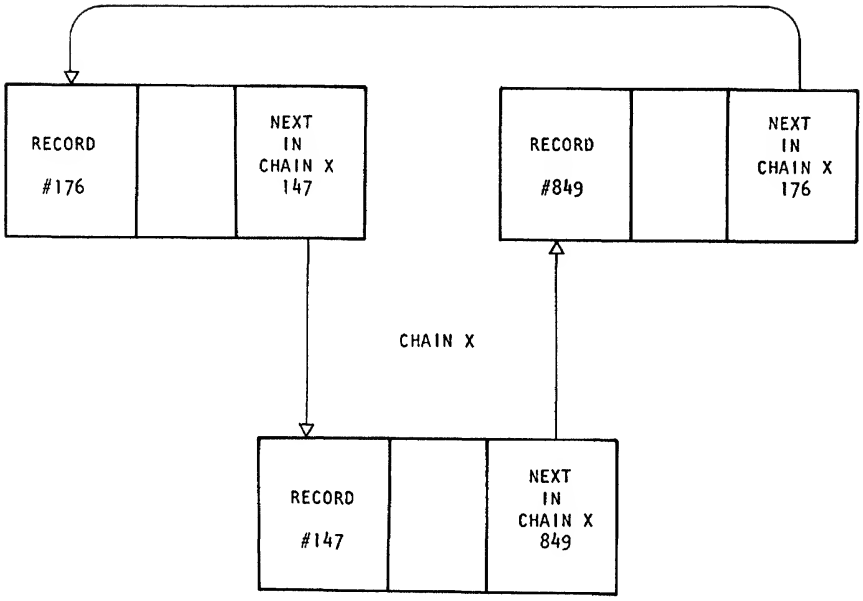


FIGURE 3 CHAIN POINTERS

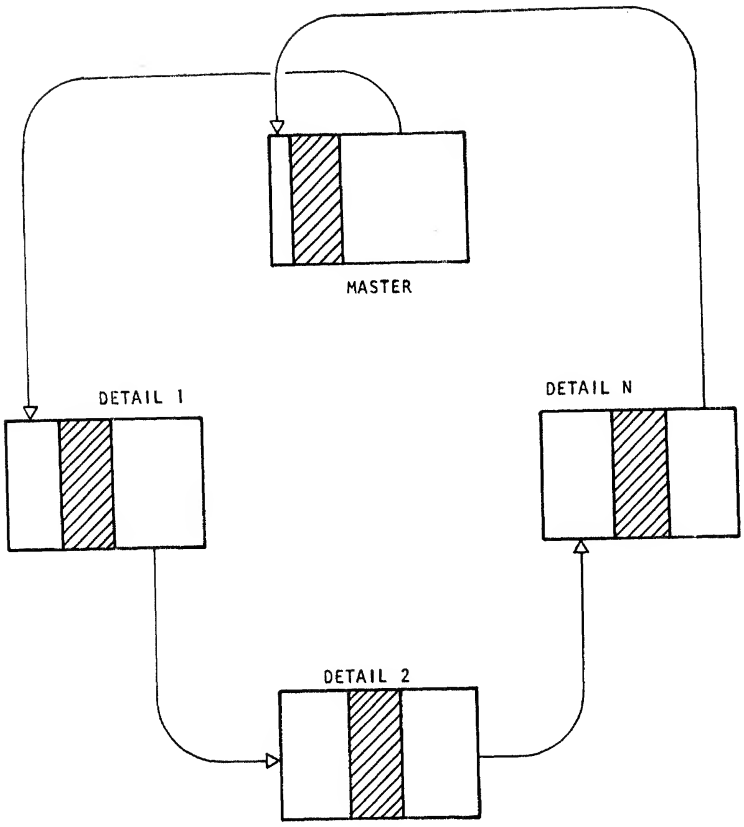


FIGURE 4 I-D-S CHAIN

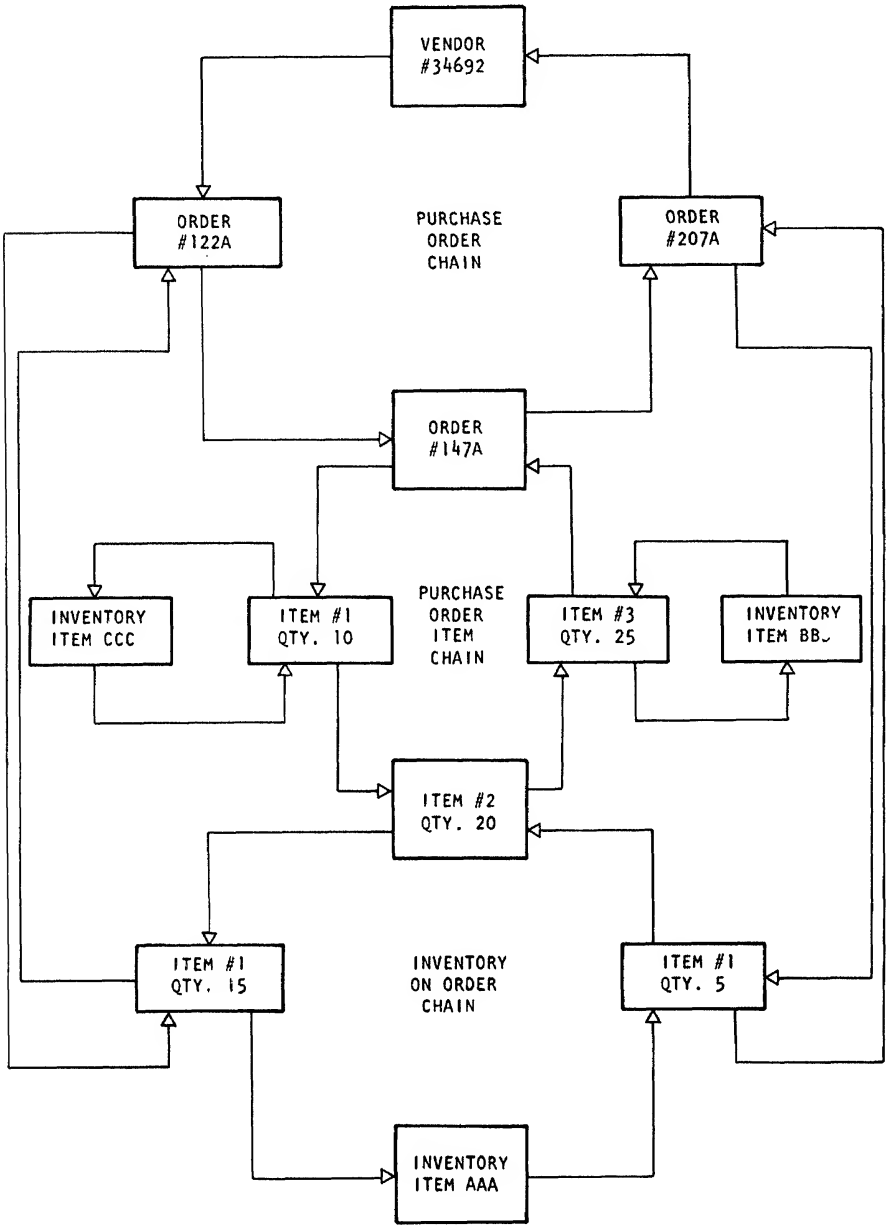


FIGURE 5 CHAIN NETWORK

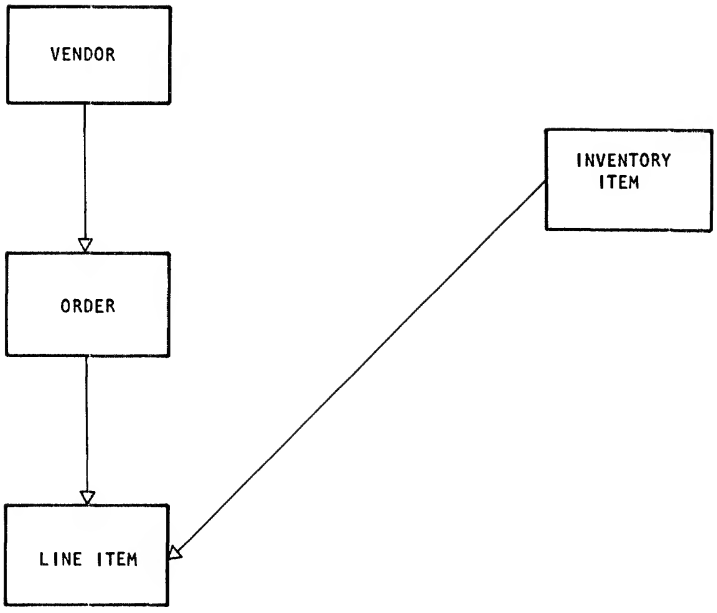


FIGURE 6 CHAIN NETWORK SHORTHAND

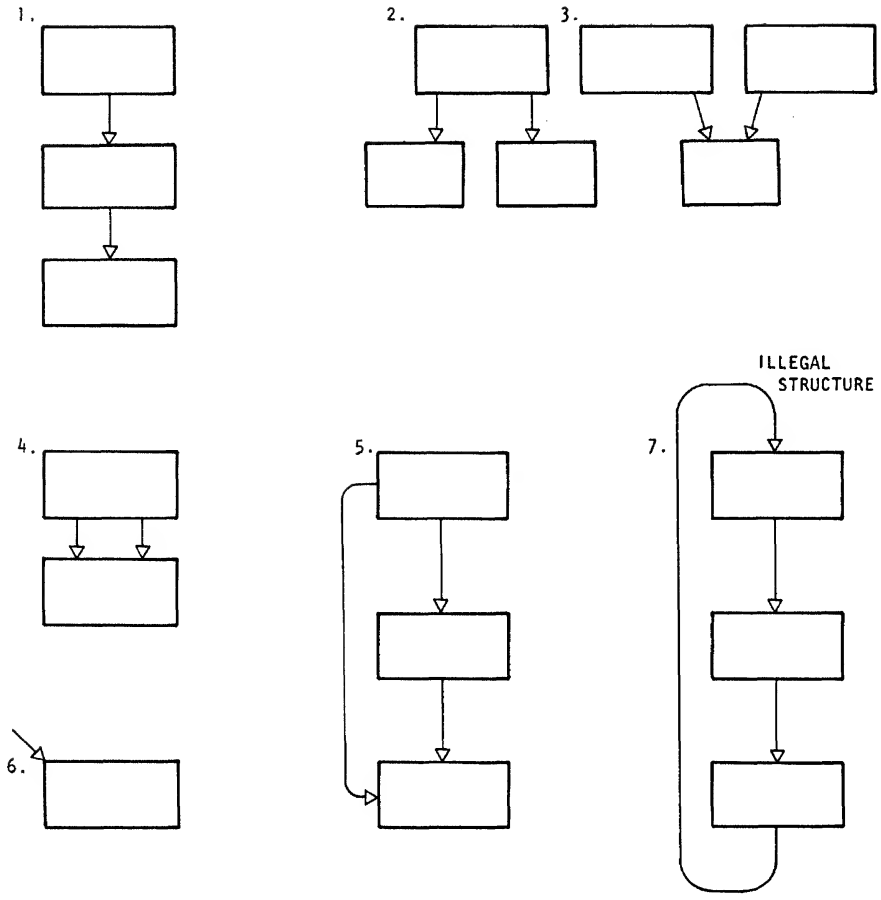


FIGURE 7 LEGAL AND ILLEGAL 1-D-S STRUCTURES

Courtesy J.J. Gonino H.I.S.

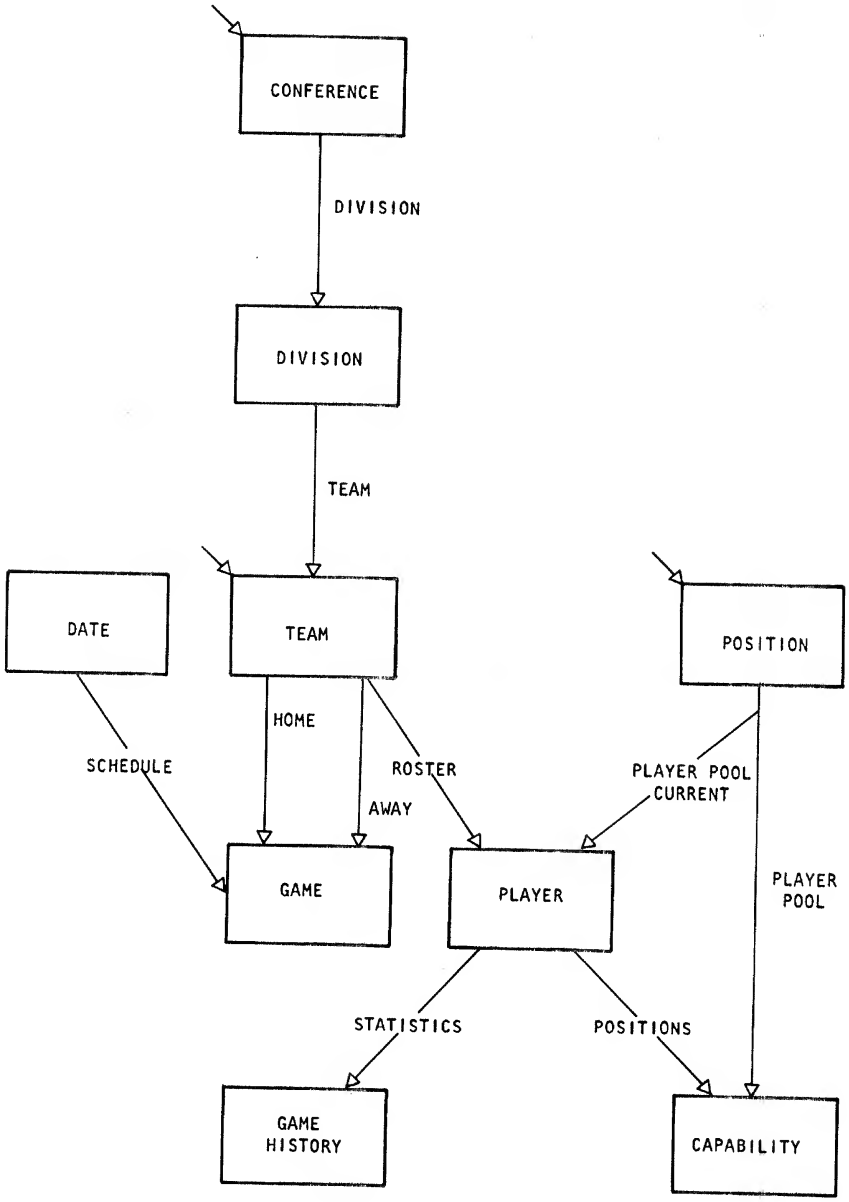


FIGURE 8 N.F.L. FOOTBALL DATABASE

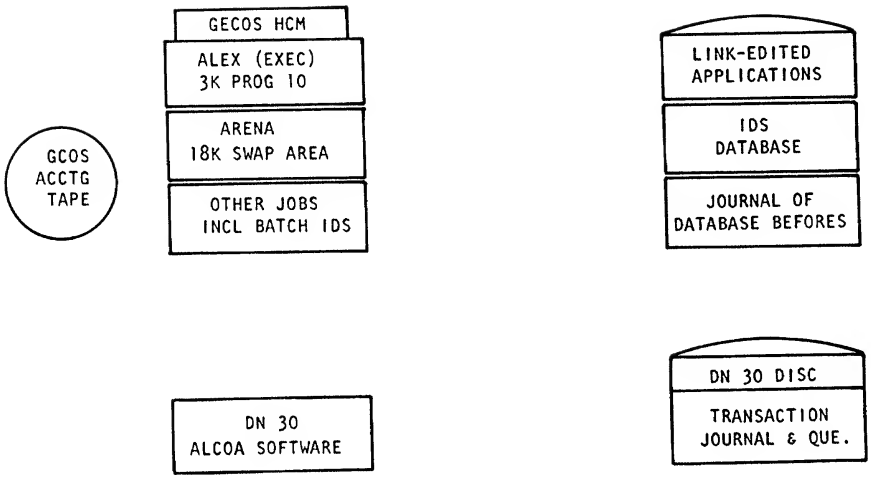


FIGURE 9

ON-LINE IDS UPDATE AND RECOVERY SYSTEM

CONCEPTUAL LAYOUT

Non Equivalences:
- debug
- complex form of delete
- error status codes
- user chain table access and updating
- sort
- retrieve next of calc chain

IDS has provided a data structure concept which has been flexible and powerful and easily represented the business system we wanted to model. My hope as a user is that the next data base product by Honeywell will have the conceptual longevity that IDS has had. Ten years is a long time in this new science.

DISCUSSION

TURNER:

Do you expect that Honeywell will now develop some of the dependency protection that existed at Weyerhauser or is that gone forever?

VONGOHREN:

I can't answer that. At our site, we have not seen the need for it yet. We are not operating in a multiple update environment.

MURRAY:

The first speaker today mentioned having to combine a teleprocessing front-end with the data base manager. The last two speakers including yourself haven't mentioned anthing about that. Can you comment on that?

VONGOHREN:

The transaction processing executive provides that capability. Although we haven't started using it yet, we find that there are many users that are talking actively and look like they're going to be using it. Where you are using your system as all things to all people, then you have so many different kinds of applications you can't begin to have dedicated on-line systems. You have to have some kind of executive and it fills that need very well from what I hear.

Data Base Management Systems, D. A. Jardine (ed.).

DMS1100 USER EXPERIENCE

E. J. EMERSON
Burndy Corporation
Norwalk, Connecticut

DESCRIPTION

Univac's Data Management System for its 1100 Series Computers, DMS1100 has been developed internally by Univac based on the CODASYL Data Base Task Group reports. In some areas Univac's implementation is, as yet, incomplete but overall it follows very closely the system specified by the DBTG.

DMS1100 consists of a data description language (DDL), a COBOL data manipulation language (DML), which has been implemented via a preprocessor which generates call statements and the Database Management System itself. In addition, a number of utilities have been provided to assist in the use of the system. The DDL compiles a source schema into an object schema which is referenced by the DML preprocessor and by DMS at execute time. The DML preprocessor accesses the object schema to obtain record descriptions and validate set memberships. However, nothing regarding the actual set memberships, number of pointers etc. is embedded in the program. Therefore, with the exception of the record descriptions it uses, the program remains independent of the database.

DATA DESCRIPTION LANGUAGE

The DDL is divided into three sections. To show the capabilities of the system I will describe each section and compare it to the Data Base Task Group (DBTG) specification.

1. The first section is the Area Section. An area is defined to be a named subdivision of the addressable storage space in the database. It may contain occurrences of records and sets or parts of sets of various types. The area section describes the characteristics of all areas of the database. The number of pages contained in each area, their size, and their division into prime data and overflow pages must be specified. The page size specified here becomes the unit of physical data transferred between secondary storage and the system buffer. An area can be specified as being expandable to a larger size in which case the maximum number of pages must be given. Before records can be stored in an area, the area must be initialized either by a utility or with a special format of the open command. An initial load factor can be specified in the area section allowing empty space to be left on pages when the database is first created. Thus, control over some physical characteristics of areas within the database is given to the data administrator in the area section rather than in a Device-Media Control language as specified by DBTG. The allocation of these areas to particular devices, which must be direct access, is done with the standard Job Control Language. The area section features which have not been implemented are: temporary area, privacy lock, and on.
2. The second section is the Record Section. A record is defined to be named collection of one or more data items. There may be any number of occurrences for each record type defined in the database. The record section describes the characteristics of each record in the database including the way in which it is stored and a definition of each data field it contains. The way in which a record is stored is specified by the location mode clause and can be direct, calc, or via set, as in DBTG. The ability to specify an 'interval' of pages has been added to the 'via set' location mode. This permits records

at intermediate levels in the hierarchy to act as cluster points for records at lower levels in the hierarchy. We intend to make use of this feature to store an employee's payroll record (one per employee) via the employee-payroll set and then to store his weekly or bi-weekly payroll history records via a set having the payroll record as owner. This allows us to have all the payroll history records for an employee physically near each other on the database. Again, this gives us an additional measure of control over the physical placement of records on the database and is transparent to the application programmer. I should point out that the location mode clause is not optional as it is in the DBTG and in some cases this has caused us to set up dummy owner records and sets so that we can store a record via set when we really have no need for the set. The record section features which have not been implemented are: on, privacy lock, actual/virtual source/result, check and encoding/decoding.

3. The third and last section of the DDL is the set section. A set is a relationship between a single owner record type and one or more member record types. The record types involved in each set and the manner in which their relationship is to be physically and logically maintained is described in this section. The owner record type and all permissable member record types must be specified. DBTG specifies that for sets of which there will be only one occurrence, the owner can be specified as 'system'. This is not permitted in DMS1100, an omission which has caused us to store one 'area-control-record' in each area of the database which we then define as the owner of any singular sets which occur within that area. The method to be used for physical set maintenance is specified in the mode clause. The only set mode currently provided under DMS1100 is 'chain' with the option to link prior, but pointer array will be provided in the near future. The logical maintenance of the set is specified by the order clause and can be first, last, next, prior, or sorted. Whenever a new member record is added to a set its logical insert point is determined by the order clause. If a record type is an automatic member of a set it will be added to the appropriate set occurrence at the time it is stored, otherwise, the record type is a manual member and an insert command must be used to add it to the set. If a record type is automatic then it is also 'mandatory'; that means it must always remain a member of the set. If, however, its membership is manual then it is also considered to be an 'optional' member and can be taken out of the set by using the remove command. The set section features which have not been implemented are: on, privacy lock, and search key.

DATA MANIPULATION LANGUAGE

The COBOL DML provides for the inclusion of a schema section in the COBOL program which specifies which schema is to be invoked for use in that program. Since sub-schemas have not be implemented in DMS1100 we define our entire database in one schema and it is invoked in all programs. The schema section defines what record descriptions will be copied into the program, which ones if any can be overlayed, whether they will be in the working-storage, common, or linkage sections and where the system generated entries should be placed. The close, delete, find, get, if, insert, modify, move, open, remove, and store commands operate as specified by DBTG. The 'order' command has not been implemented.

One important area in which DMS has taken an approach substantially different from that specified by DBTG is Concurrent Updating. To support this approach, the functions performed by the Keep and Free commands have been redefined and the Impart and Depart commands have been included in the DML. The Impart command must be executed before a run-unit can perform any other DML command. The Depart command must be executed by a run-unit after it has completed all other database processing.

The Impart and Depart commands define the life of a run-unit during which the following records are locked out from use by any concurrent run-unit:

1. Any record which has been altered
2. Any record for which a KEEP command has been issued
3. The current of run-unit.

A run-unit can release its hold on these records by issuing a FREE command. In conjunction with record lockout DMS1100 provides the facility to 'rollback' the updates (since the last Impart or FREE) made by a single run-unit, either at the request of the run-unit or automatically due to a program abort or deadlock detection, or by all run-unit which were active at the time of a system failure. Whenever one of our application programs detects an unexpected error the program requests rollback and the database updates made by that run-unit are undone. We have also used this facility to make it possible to restart a long batch update run after the successful completion of each transaction by keeping track of the number of transactions completed in a control record and issuing a FREE command after each one.

The danger is using the lockout approach is that deadlock may occur. When a deadlock condition is detected by DMS it uses the rollback facility to undo the updates of the run-unit which has updated the least number of records and permits the others to continue. The program which was rolled back is notified of this condition and may restart its processing.

IMPLEMENTATION

Unlike many other users of Data Management Systems, our database is not large enough to have forced us into the use of such a system. We were in the position of trying to support a variety of systems which had developed over the years with many non-integrated files containing a great deal of redundant data and producing results which were frequently inconsistent.

We knew that we could not continue to operate these systems as they were but we wanted to be sure that as we converted them and cleaned up the data they contained we made it convenient for new systems being developed to take advantage of the work which was being done. This required that we have a reasonable amount of program independence from the database since we were not at all sure what the final database would look like and we did not want to have to do extensive maintenance to existing application programs whenever we added something new. In other words, we wanted existing systems to become as asset rather than a burden.

I think we have been fairly successful in achieving this goal. We first converted our inventory control system to DMS1100, putting all existing information for item description, inventory, and requirement status on the database. We then added product structure and routing information and most recently we have added all the necessary record types to support an on-line Order Entry system. Thus, we have evolved our database and have found that as we added new information we had to make very few changes to existing database application programs.

Cleaning up our existing data to make it usable by other systems has been a far more difficult task than we expected. The logical structure of the database, which was designed to eliminate inconsistency, has forced us not to accept errors which we might otherwise have tolerated during the conversion. For example, the old order entry system had an open order file which divided each order into 'parts' for shipping purposes and carried the customer's account number and bill-to name and address for each part of order. Since each order was for a single customer these fields should have been identical for all parts of order but frequently they were not. Since this problem would be eliminated on new orders once the database was built and the incorrect data would be eliminated as the old orders were shipped, putting this bad data into the database would not have been intolerable. However, the structure we had set up, largely to eliminate this problem of inconsistency, would not allow us to compromise.

Now that most of the old data has been converted we are beginning to get a real payback from the database system in terms of our ability to respond to requests for information from user departments within the company. The time it takes to produce the answer to a question is now fraction of what it was before.

Now that most of the old data has been converted we are beginning to get a real payback from the database system in terms of our ability to respond to requests for information from user departments within the company. The time it takes to produce the answer to a question is now fraction of what it was before.

The current database has 34 million characters of user data and 11 million characters of system overhead data. It is spread over the equivalent of 4 2314 disks allowing a substantial amount of room for expansion. As shown in the accompanying diagram the database consists of 21 record types and 23 set types which are divided into 8 areas. The number of occurrences of each record type is shown next to its box on the diagram. To show the kinds of structures that can be supported by DMS I would like to discuss some of the things we have done.

The product-structure record is a member of two sets: the component and where-used sets, which both have the same owner record type. Since we desired to use the location mode of owner for both sets we used the Alias option on the set occurrence selection clause for the where-used set. Since there was the danger of a cycle being introduced into the product-structure through a user input error we wrote our own code to update low level codes and check for them. If a cycle is encountered we use the rollback feature to undo the low-level code updating and store of the product-structure record.

Unfortunately, we have two different identifiers for all of our products. One known as an IBM number is a six character numeric identifier which is used internally by some departments. The other known as the catalog number appears in the company's catalog and is used internally by other departments. In order to be able to access an item record by either identifier we have declared the IBM number as the calc key for the record and we have established a 'catalog-cross-reference' record having a calc key of catalog number. For a number of reasons unique to our application we have chosen to implement this using a sub-routine which the application calls when it wishes to obtain an item record using the catalog number; but we could have done it using a set in which case a DML command could have been used to obtain the record based on either key.

In our order entry system we have made use of several manual sets for keeping track of the status of open orders. When entry of an order is begun, a sales order header record is established and it is inserted into the manual 'orders-in-entry-set'. Parts-of-order are then established for each different set of shipping instructions specified by the customer and the line items are entered for each part. These individual customer requirements become members in the automatic 'order-detail' set. When a complete order has been entered and all items validated and scheduled, each customer requirements is inserted into the 'requirement-order' set and the order header is removed from the 'orders-in-entry' set and placed in the 'orders-awaiting-shop-papers' set. This set is then read by a nightly batch run which prints shipping papers for each job, removing orders from this set as the printing is completed. Two other manual sets, 'orders-on-credit-hold' and 'orders-of-interest' are also used for keeping track of open orders for which special action is required. When a shipment is made the customer-requirements involved are removed from the 'requirement-order' set and inserted into the 'shipment' set thereby becoming receivables. Thus the original 'customer-requirement' record serves several functions as determined by its logical set membership without physically moving from its original place in the database.

CONCLUSION

There are currently 15 DMS1100 users in live operation and another 20 to 25 who are in the development phase. One of the production users and several of those who are in development have databases of 1 to 5 billion characters with the other installations having from 10 to 200 million characters of data. We have found DMS1100 to be a flexible and easy to use system and have been extremely satisfied with the results which we have obtained.

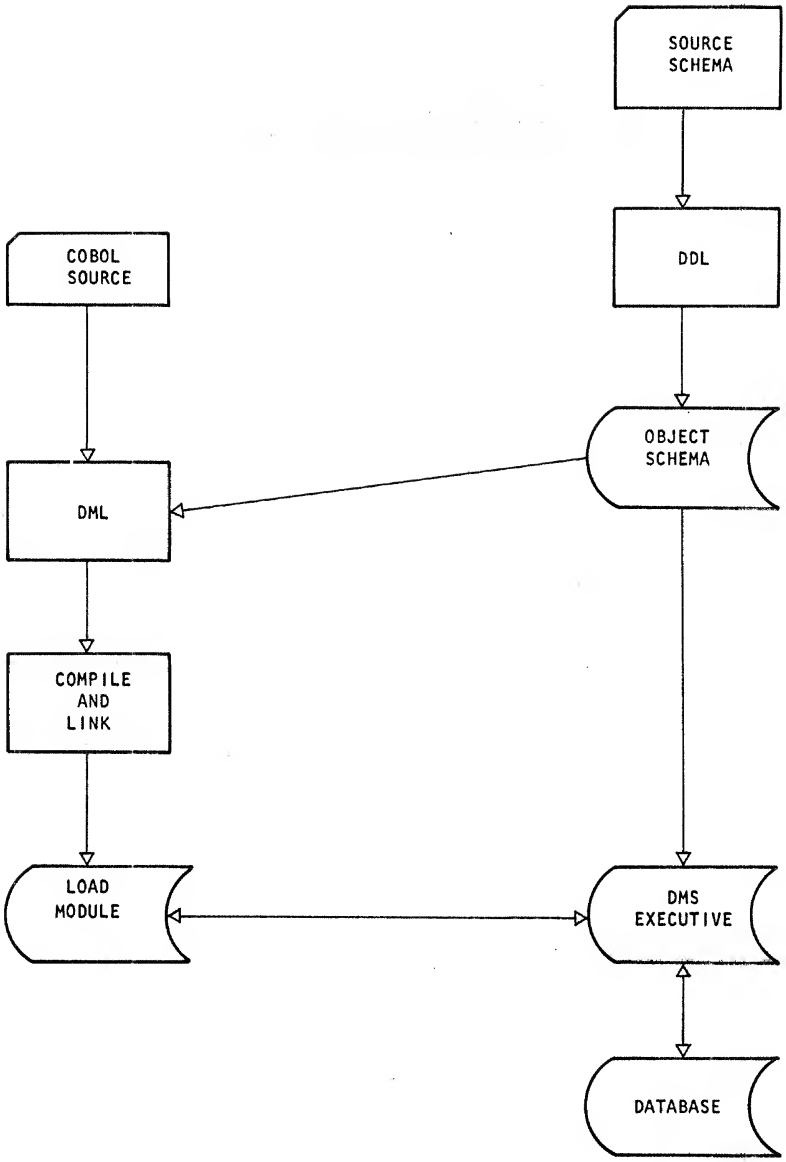

SOURCE SCHEMA
DDL
COBOL SOURCE
OBJECT SCHEMA
DML
COMPILE AND LINK
LOAD MODULE
DMS EXECUTIVE
DATABASE

FIGURE 1

DATA DEFINITION LANGUAGE (DDL)

1. AREA SECTION
2. RECORD SECTION
3. SET SECTION

DATA MANIPULATION LANGUAGE (DML)

	CLOSE	INSERT
	DELETE	KEEP
*	DEPART	MODIFY
	FIND	MOVE
	FREE	OPEN
	GET	REMOVE
	IF	STORE
*	IMPART	

* New Commands

FIGURE 2

Use of the Interval Clause with Via Set Location Mode

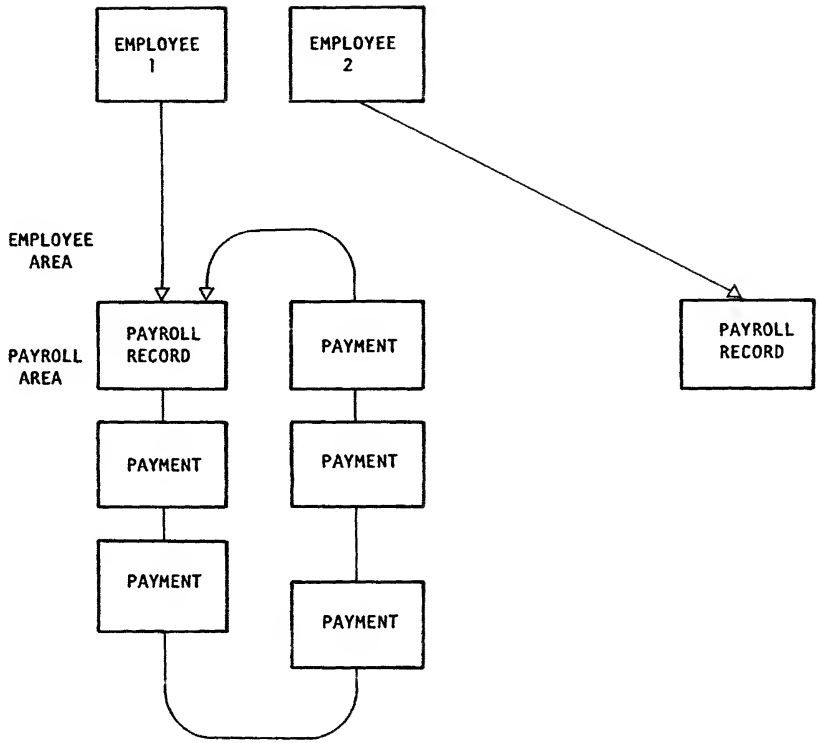


FIGURE 3

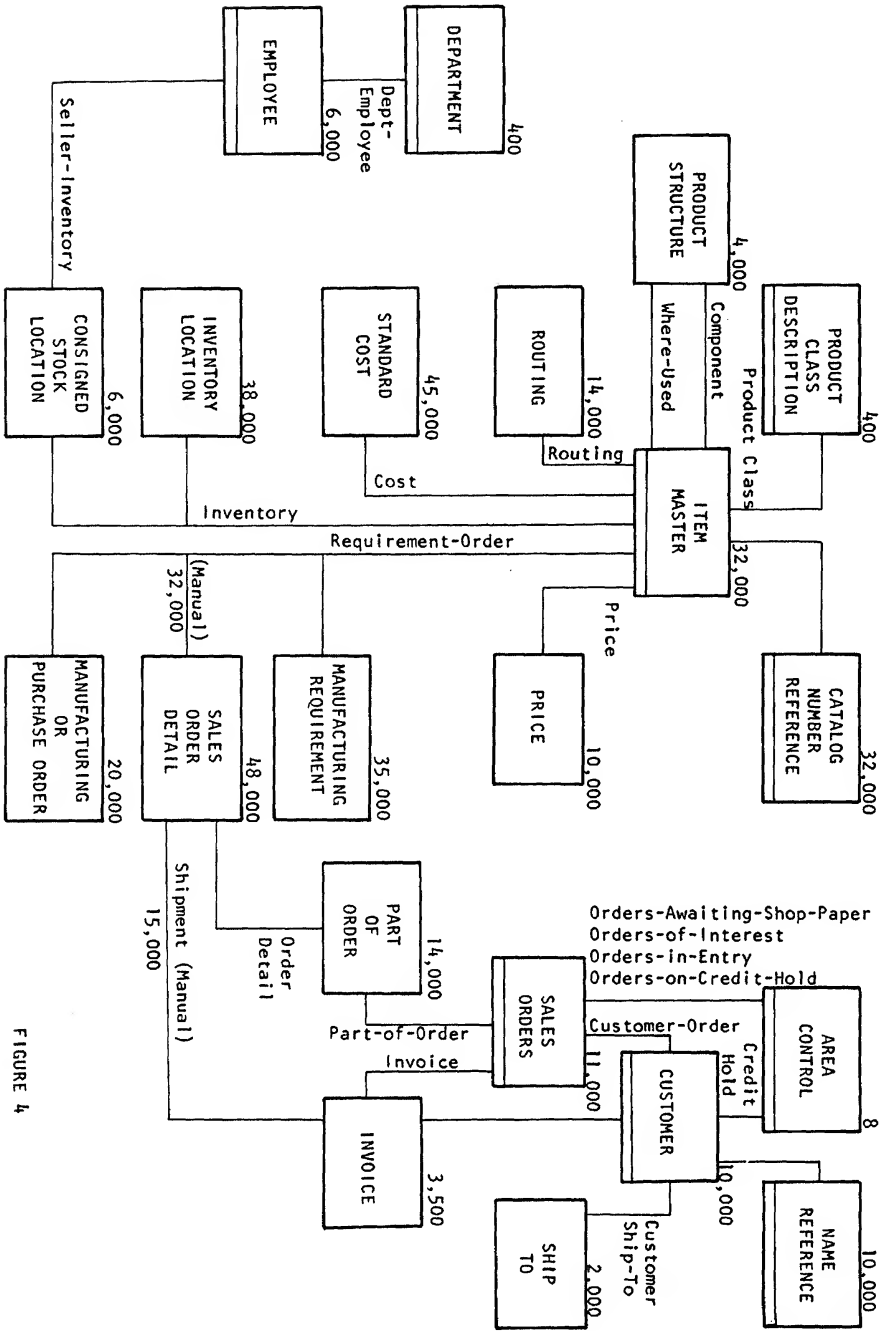

EMPLOYEE
DEPARTMENT
Dept-Employee
6,000
400
Seller-Inventory
PRODUCT STRUCTURE
4,000
Component
Where-Used
PRODUCT CLASS DESCRIPTION
400
Product Class
CONSIGNED STOCK LOCATION
6,000
INVENTORY LOCATION
38,000
STANDARD COST
45,000
ROUTING
14,000
Routing
Cost
Inventory
ITEM MASTER
32,000
Requirement-Order
(Manual)
32,000
Price
MANUFACTURING OR PURCHASE ORDER
20,000
SALES ORDER DETAIL
48,000
MANUFACTURING REQUIREMENT
35,000
PRICE
10,000
CATALOG NUMBER REFERENCE
32,000
Shipment (Manual)
15,000
PART OF ORDER
14,000
Order Detail
Orders-Awaiting-Shop-Paper
Orders-of-Interest
Orders-in-Entry
Orders-on-Credit-Hold
SALES ORDERS
11,000
Customer-Order
Part-of-Order
Invoice
CUSTOMER
10,000
Credit Hold
AREA CONTROL
8
INVOICE
3,500
SHIP TO
2,000
Customer Ship-To
NAME REFERENCE
10,000

FIGURE 4

DISCUSSION

JARDINE:

You mentioned that DMS1100 has a method for detecting deadly embrace. Is this a system function or does it merely detect the existence of locked records and pass the problem back to the application program?

EMERSON:

It is a system function. If a program requests a read of a record which is locked by some concurrent run unit, that run unit is queued until that page becomes available. An Impart command must be issued before any other command can be issued. Whenever any run-unit is queued waiting on a locked page, no other run unit can Impart. If a run unit attempts to Impart, it is queued waiting until these locked pages are resolved. If there are no active run units which are not queued, then that is considered to be a deadly embrace. Since no more active run units can come into the system, when all run units that were active have departed, and the only ones remaining are those that are queued waiting on locked pages, that is considered to be a deadly embrace condition. That solution is all right as long as you don't have any long running batch tasks at the same time, such as database maintenance programs. Probably some more can be done in the area of deadlock detection, but that method will work as long as we are careful what we run with our on-line system. When the deadlock is detected, a rollback mechanism is used to undo the update of the run unit which had updated the least number of pages. Control is then returned to a predetermined point in that program. The programmer knows that he has been rolled back because of the deadlock situation, and he can restart his processing.

JARDINE:

Have you any way of measuring how many deadly embraces that you get per month?

EMERSON:

There is no notification to either system programming or to the operator. The application program itself is notified and we could keep track of it but have not. We are not really in heavy enough multiple update use to give you an accurate figure on the deadly embrace occurrence rate.

STOCKHAUSEN:

You mentioned that you used a subroutine to translate between your various numbering schemes. You could have this through the set description capability. is there something inherent in the description facility or the implementation you are using that made you choose this way?

EMERSON:

The sales department insists that dashes, slashes and blanks in the catalog number are not relevant and we are supposed to take them out before we do this translation. UNIVAC has not implemented the ON command so we weren't able to use that. We also wanted to provide facilities to reference the same items by different stock number or by different catalog number. We sometimes want to reference by the military spec number, or even some other company's catalog number in the case or products which we sell mostly to one another company. We could have done that by a set, but we would have to fetch the member record and then fetch an owner which takes two commands anyway. For those two reasons we prefer to do it by subroutine call. I think this really must depend on our application rather

than on any limitation in DMSII00 itself.

STOCKHAUSEN:

You are saying that you are going to a database system so that you can take the meaning that is embedded in the data and factor it out. Now you are saying that you are reversing the process: that you are going to code a subroutine to take the meaning in the data and tuck back on into the programs where it is once again obscure. Isn't this a reversal in your development?

EMERSON:

Well, it's in the subroutine.

STOCKHAUSEN:

Is there a problem in the system that forces you to do this?

EMERSON:

No, there is not.

STOCKHAUSEN:

Is this just expedience?

EMERSON:

There is not a problem in the data management system. We choose to do that because of certain problems that we wanted to get around in this way. I agree that we are in fact putting it back into the application program, but we have put it in a subroutine and I expect that when the ON command gets implemented we will be able to put it in a database procedure rather than in an application program. Then I think we will be able to go back to using a straight fetch command.

STOCKHAUSEN:

What happens if a program ceases for some reason, say a program check, before it has freed up things; that it expires with exclusive control on. Suppose it has added to the structure. Is there some structural fault in your data base? Do you go back and make things look all right? What is the status of your database?

EMERSON:

The system catches that contingency and it rolls back the offending run unit. Its updates are in fact undone back to its last Impart or Free command.

HILL:

Can you have multiple Imparts and departs per run unit effectively creating subrun units?

EMERSON:

Yes.

HILL:

So that in fact you can rollback portions of a run unit?

EMERSON:

You can only rollback the most recent portion, back to the most recent Free or Depart command. You can't divide it up into sections and then say you want to rollback current Depart minus 3. You can however divide it up into sections. That is, the function of the Free command. You need not do a Depart and Impart; you can issue a Free and that releases all the locks you currently have. If you get into trouble you go back to that point. We have used that facility in one long batch run that does most of the updating. We have used the Free command after each group of transactions to allow restarting at any point in the middle.

HILL:

Do you plan to use this technique for on-line processing?

EMERSON:

Yes.

GIBB:

Do you know how much time is used to maintain the audit trail necessary for recovery and rollback. Also, how often have you found that you have to recover using the audit trail?

EMERSON:

Audit trail is different from rollback. It is an available facility which we have not used yet.

GIBB:

So you recover entirely using the rollback?

EMERSON:

Yes. If we get head crash or some physical problem, we would have to restore a previous version of the database and reprocess the transactions.

GIBB:

You must maintain some record of what happened in order to rollback?

EMERSON:

We must keep track of what has been processed against the database since we took the last physical dump. There are two recovery facilities, one of which is called rollback which concerns itself with what UNIVAC terms quick looks. This is an image of a physical page which is written to disk prior to performing the actual update. Those are used for rollback. They also have an audit trail which writes the before and after image of the database pages to audit trail tape, in addition to writing check points indicating what units were active at a given time. We use the quick look facility. I cannot answer your question regarding physical accesses as a result of quick looks.

GIBB:

Have you ever had to recover by using a database dump, and does this happen frequently?

EMERSON:

Yes, but not often.

GIBB:

Can you estimate what congestion occurs as a result of lockout?

EMERSON:

A statistic that would be of interest in that regard would be how many run units are waiting on a page held by another run unit. I don't have the facilities to measure this. They are available in a different version of the preprocessors than the one that I'm permitted to use because of the compiler we're using.

LOWENTHAL:

Are you using the multi-thread version? How much code does that require?

EMERSON:

It takes 30 K words which is roughly equivalent to 150 K bytes. It doesn't take more than a single thread version, in fact it takes less if you have more than one person running it at a time. With the single thread version, the entire data management system was linked into each application program; now there is just one copy of it and it's shared among the active users.

LOWENTHAL:

Is there any variable core sizes?

EMERSON:

Yes, it's 20 K for the actual code and we allocated 10 K for buffers. You can allocate down to 4 K buffers or up to the size you want.

TURNER:

Do you make use of any of the security provisions and if so can you very briefly give us some of your experiences?

EMERSON:

We make use of every one. That is, there aren't any.

Data Base Management Systems, D. A. Jardine (ed.).

DMS APPLICATIONS AND EXPERIENCE

P. A. LAVALLEE and S. OHAYON
Xerox Data Systems
Research Laboratories Division
Rochester, New York

This paper is organized in five sections. The first section gives a broad brush description of a Data Management System (DMS) which is closely related to the DBTG recommendations, implemented and offered as a fully supported product by Xerox Data Systems. The next three sections outline three applications which have been built with this DMS system (data base, data definition, structure diagrams and sample transactions are included). In the last section, several conclusions are drawn about the value and drawbacks (leading to further research avenues) of DBTG like Data Management Systems.

1. INTRODUCTION TO DMS-E

1.1 Extended DMS provides a capability for accumulating large volumes of data into a single data base, which may be structured to reflect any desired data relationships. The Extended DMS system is offered as a fully supported product program by Xerox Data Systems. This system is composed of thre main sub-systems which are:

1.1.1 DMSFDP (File Definition Processor) which takes the description of the data base as specified in a Data Definition Language (DDL) and generates a schema file and a sub-schema file. The schema file describes the complete data base, its size, record types, record storage and retrieval techniques and attricutes, privacy controls, etc. The sub-schema file describes the complete data base or just those portions that are required by a specific application.

1.1.2 DBM (Data Base Manager or in DBTG terms, the DML or Data Manipulation Language). This is a library of routines for accessing and updating a DMS data base. It consists of various kinds of Find, Store, Delete or Modify routines, error processing routines, journalizing, training and statistics collection routines.

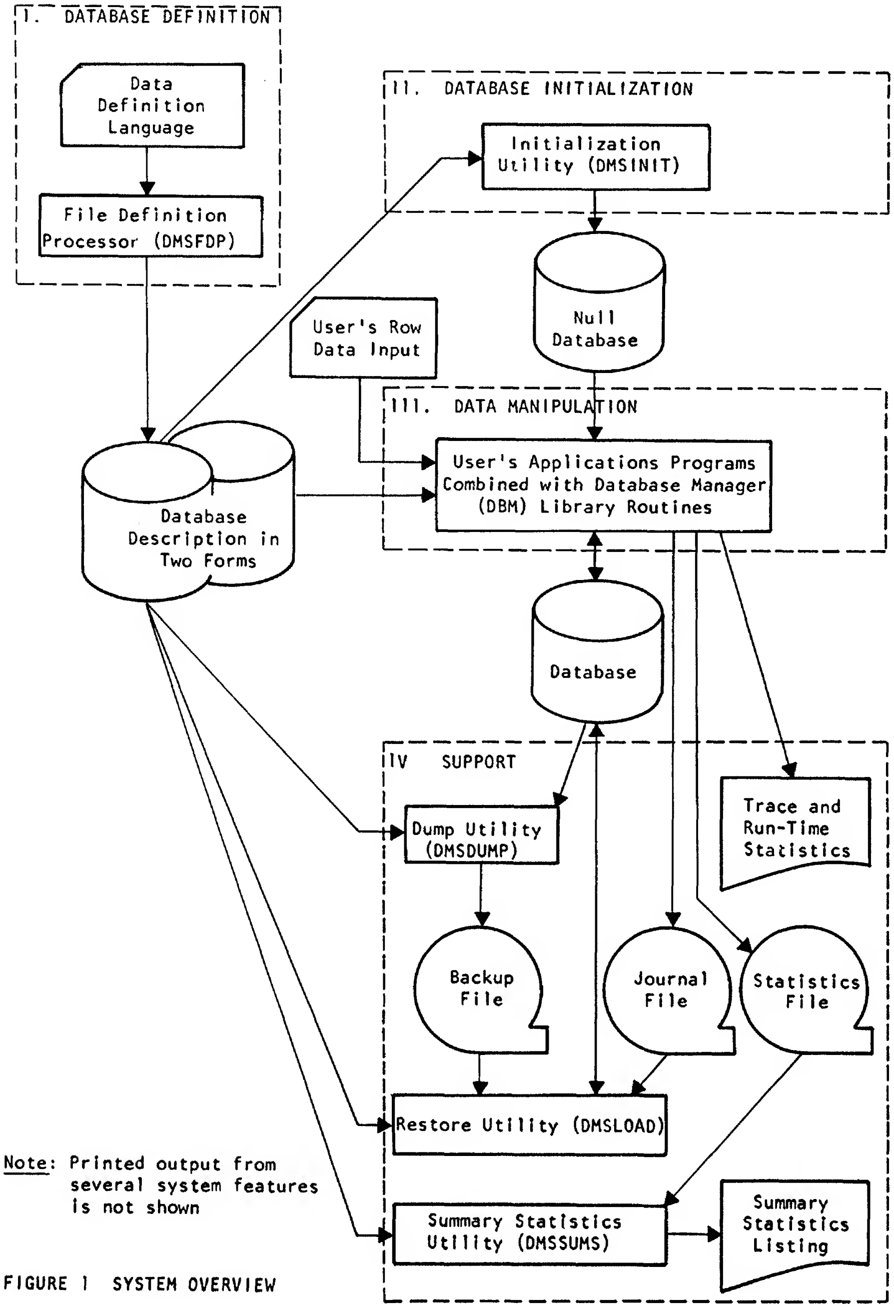
1.1.3 Utility Processors - several utility processors are provided whose functions are: (1) initializing areas before any data is stored; (2) dumping selected or total contents of an area and saving it for backup; (3) updating the saved data with journaled pages for recovery purposes; (4) printing selected portions of an area, journal or backup file for visual checking; (5) printing summary statistics collected by the DBM into a statistics file.

A system overview is given in Figure 1 as depicted in Xerox (1972).

1.2 An overview of the DDL, DML and Utility is now given, and those familiar with the DBTG Report will notice the close relationships; for those not familiar with the syntax and semantics of generalized Data Management Systems, we recommend Xerox (1972) and CODASYL DBTG (1971).

1.2.1 DDL Overview - The DDL consists of five types of entries; each type is functionally described. The exact syntax can be obtained in Xerox (1972).

a) Schema Entry
Schema entries supply the file name for the schema file and specify locks and passwords for limiting access to the schema itself and to the user's data base.



Note: Printed output from several system features is not shown

FIGURE 1 SYSTEM OVERVIEW

b) Area Entries
Area entries supply (1) the file names by which the data base areas are identified for the host operating system and in the user's working storage declarations generated by the FDP; (2) information on the size of the area file; and (3) information on how the file space is to be managed by the DBM.

c) Group Entries (DBTG Records)
Group entries specify the size, form, and order of appearance of item values within group occurrences, the method for locating occurrences, the privacy locks that are to control access to the occurrences, and which items, if any, are to serve as secondary indexes. Corollary groups or subgroups, used to manipulate secondary indexes, are defined to designate items as secondary indexes.

A group entry consists of a group subentry, followed by item subentries for all items in the group, followed by invert subentries for all of the corollary groups that control secondary indexes for the main group.

Group Entry Skeleton

```
Group subentry

First item subentry
    .
    .
    .
Last item subentry

First invert subentry
    .
    .
    .
Last invert subentry
```

Group Subentry
Group subentries specify the name of the group, the criteria for identifying a specific group occurrence, the guidelines for placing the group in physical storage, and the privacy controls for the group.

Item Subentries
Item subentries specify the characteristics of the items in the group. All of the item subentries for a group together provide an image of the data portion of the group occurrences in the data base.

Invert Subentries
Invert subentries identify the items in the group that are to serve as secondary indexes, providing an alternative technique of identifying specific group occurrences for retrieval. (The primary technique is determined by the group's location mode.)

d) Set Entries
The set entries define all the user-specified relationships among group occurrences by indicating which group types are to participate in which sets, what set pointers are to be included in the group occurrences, what is to determine which owner group occurrence a particular member group occurrence is to be associated with, and how the member occurrences are to be associated with each other.

Set Entry Skeleton

```
Set Subentry
Member subentry
(Member subentry)...
```

Set Subentry
Set subentries provide the name by which the set is referenced in other DDL entries (e.g., definition of groups whose location mode is via set, and for subschema selection).

Member Subentries

Member subentries identify the groups that are to be members of the set and specify all the controls that are to apply when a new occurrence is stored or whenever a member occurrrence is linked into a set occurrence.

e) End Entry

The last entry in the definition of a schema must be END.

1.2.2 DBM - The Data Base Manager is the term applied to the library routines that are used with a user's program to store, retrieve and update a data base. The following list of statements indicate the types of operations (user procedural calls) available to the data base administration staff. See Xerox (1972). These are grouped into 10 subsets.

a) Beginning of Processing

Before any data manipulation activity can occur, the files in which the data is stored must be opened. The DBM interacts with the operating system to open the file in response to an open-call from the using program. The open-call not only identifies the area to be opened, it also indicates what type of activity is intended.

b) Adding Occurrences

The first activity involving the data in the data base is to load, or store, group occurrences in the area files. This activity continues with varying frequency over the life of the data base.

c) Deleting Occurrences

A group occurrence can be physically removed from the data base or marked as unavailable and flagged for future removal or the delete call can specify conditions under which the group is to be deleted.

d) Modifying Data Values

The values of one or more items in a single group occurrence can be modified.

e) Modifying Linkages

An occurrence of a group whose membership in a set is defined as optional or manual can be linked to or delinked from a set occurrence (LINK and DELINK). Also, a member group occurrence can be changed from one owner occurrence to another in any set in which it participates (RELINK).

f) Retrieving

Various techniques are used for retrieving specified group occurrences from the data base and making them available to the user. The selection of the technique depends upon the specific application. Technique selection must be governed by the group and set characteristics of the occurrence to be retrieved.

g) Run-Time Statistics

To initiate and terminate, by calls to the DBM, the collection of statistics on the performance of a program as it accesses a data base. The statistics reflect the activity of that job only.

h) Error Control

To enable the user's program to maintain a degree of control over the handling of DBM-detected errors by issuing a call that specifies a location to which the DBM is to return control in the event of a specified error condition.

i) Checkpointing

To add an additional protection to the integrity of the data base by allowing the user's program to periodically request that the DBM write all modified pages to the data base.

j) Terminating Processing

To close opened areas when a program's data base activity is finished.

1.2.3 Utilities: The utilities are usually parameterized with area names, page ranges, cipher keys and group names where necessary for statistics and printouts. There are four main utilities to date. These are:

a) DMSINIT: utility which initializes an area or areas of a data base, or specified pages in an area.

b) DMSDUMP: utility which dumps either all or selected parts of existing data base areas to a sequential file or a printer.

c) DMSLOAD: utility which restores all or part of existing data base areas from a sequential file or tape or disk.

d) DMSSUMS: utility which prints the total contents of the statistics file generated by the DBM or selected counts from the file. The user may select area counts, group counts or set counts for all specified areas, groups, and sets by means of statistics selection input paramaters.

We now proceed to some applications.

2. ARCHIVE CONCEPT - A MULTI-LEVEL FILE SYSTEM USING DMS

2.1 ARCHIVE Revisited (see Crean (1972), Lavallee (1973).

This section briefly describes the ARCHIVE file storage system and outlines the processes involved in user interaction with his files stored in the ARCHIVE. A full description of the system was presented at the December, 1972, and June, 1973, XDS Users' Exchange meetings.

ARCHIVE is a major file system enhancement to UTS (Xerox 1971)). The unit of transaction is the file - record access or modification within the ARCHIVE is not performed. It is a segmented, self-cataloguing structure, implemented through the use of DMS and the Private Disk Pack features of UTS.

The ARCHIVE concept was developed for what we refer to as a scientific time-sharing shop. By this we mean a large Sigma 6, 7 or 9 with a thousand accounts of which 50 to 100 are active in a given day. The work load includes extensive use of source file operations and compiler activity; data files are moderate in size. When a file is accessed in a day, it is in general used many times; less than half of the files accessed are altered in a given day. However, every one of the 1,000 users expects immediate access to his files. High value is placed on file stability and long-term storage (over 30 days). By time-sharing shop, we do not mean that most of the execution is on-line, but rather that the computer activity is dynamic and unpredictable. Although most jobs are initiated on-line, the significant share of the CPU activity is in servicing the batch stream.

In such an environment, there is a clear cleavage between file storage and file usage; file storage is further divided into short and long-term storage. The multi-level ARCHIVE file storage system reflects this. File management is thus separable into three areas and each can be individually optimized. Figure 2 illustrates this hierarchy and the user's interactions with this file system.

a) File Usage

File usage is handled by the usual public storage system in UTS. It need only be moderate in size (about 1 pack to handle daily use); half a rapid access drum is sufficient to handle file directories and temporary print files.

It is:
- easy to use
- efficient in space allocation
- rapid in file access
- designed for file update and creation
- interfaced to all processors

b) Active File Storage

File storage is maintained in a set of private disk packs called the ARCHIVE. Each disk is an independent (of any other ARCHIVE pack) DMS data base containing user files; only the highest level of the account directory is repeated on each pack. The user retrieves and inspects his files directly through a user-executed processor called ARCH. To use a file, the user must fetch it into public storage via the ARCH processor. Requests to alter the user's ARCHIVEd files are received by ARCH and passed to a single ghost job. ARCHIVE storage has the following properties:

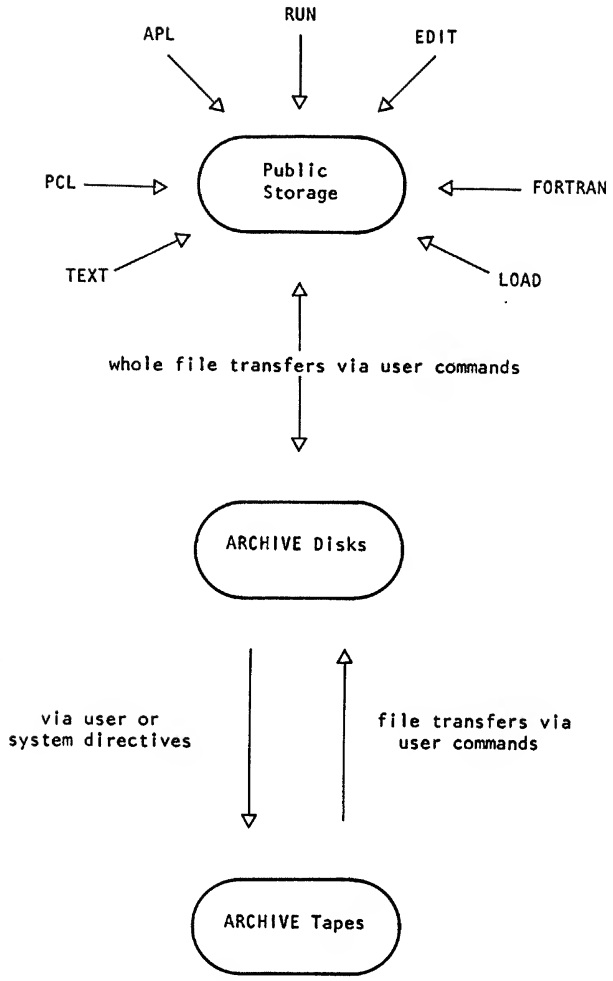


FIGURE 2 HIERARCHICAL STORAGE

High file reliability
High immunity to hardware and software errors
Extensive error detection
Isolation of errors
Volume independence
Simple, clean interface to users, public storage and the monitor
Extensive statistics gathering tools

c) Storage for Inactive Files

The third level of storage is a set of tapes containing files which had lain dormant in the ARCHIVE for some time. At intervals, these files are moved to tape and their directory records modified to indicate their location. When such files are requested by the users, these files would be copied into ARCHIVE from tape and then into his public account, automatically within 5 to 30 minutes. The time for retrieval is dependent on operator response and the queue for the tape drive.

The relative amounts of the files to be maintained on tape, the length of dormancy at which a file is migrated to tape and the frequency at which tape migration is performed are operational parameters controlled by the data base administrator.

Figure 3 summarizes the characteristics of this multi-level user file storage system. The distinctive characteristics of each type of storage can be clearly seen.

The user interacts with his ARCHIVE files through a single processor, ARCH. The basic commands relating to the multi-level data base are illustrated in Figure 4.

A multi-level file system provides many benefits:

- public file space needed for work is decoupled from file storage
- better security for stored files (not one file lost in 12 months of operation)
- high tolerance of hardware failures. The system can operate without one or more ARCHIVE packs. Critical activities like updating the ARCHIVE may be postponed while regular work continues.
- ARCHIVE can support larger amounts of user file storage than would be feasible under regular file management alone.
- tape migration provides both catalogued long-term storage and a method of automatically eliminating garbage files from the ARCHIVE disk data base.
- reduced file maintenance time. No full saves of tape restores are performed. All file maintenance jobs run concurrent with user activity. Physical dumps of the ARCHIVE packs to tape (DMS-DUMP utility) also check the basic integrity of the data base.
- higher performance from public storage with only active directories maintained on RAD.
- rapid backup of altered files. Upon user's request, a file is copied into ARCHIVE within less than 15 seconds under most circumstances.

The added costs of an ARCHIVE system are:

- the additional I/O to move files into public storage (and sometimes back into ARCHIVE).
- the user inconvenience of an extra file management processor.

2.2 ARCHIVE Data Base

Crucial to the ARCHIVE concept is a crash tolerant data base. The elements of the data base, its structure and design are elaborated upon in this section. DMS was chosen as the implementation vehicle and was found to have almost all of the characteristics required for a high reliability system. The data base consists of 'n' identical data bases, each one occupying a disk pack. Some of the design philosophies were:

	Public Storage	ARCHIVE Disks	ARCHIVE Migration Tapes
file retrieval time	immediate	several seconds	5-30 minutes
file creation time	immediate	1 minute	1 week
file inspection	immediate	immediate	summary only
file alteration	flexible	controlled	none
security	fair to good	excellent	excellent
space allocation	independently controlled by accounts		
file loss mechanism	crashes, file inconsistencies	disk destruction	tape destruction
file recovery mechanism	reload from ARCHIVE	restore from DUMP tape (24 hours old max)	use other tape copy
hardware	7232, 7242, etc.	many 7242's	many reels of mag tape

FIGURE 3 TRI-LEVEL DATA BASE

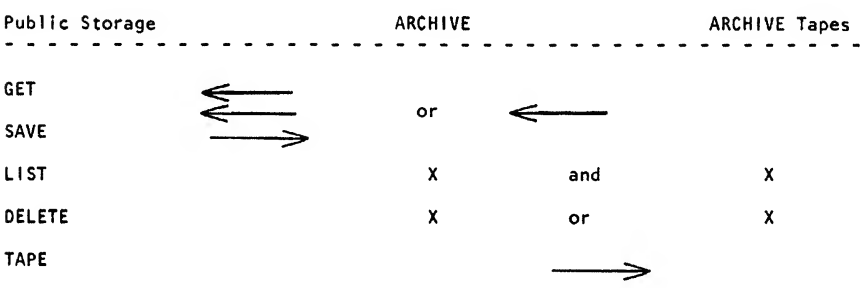


FIGURE 4 !ARCH - USER INTERACTION WITH ARCHIVE

a) all files belonging to an account were segregated into a single pack.
b) file data blocks are self-identifying in addition to being linked to a file header record.
c) all file header records are self-identifying, in addition to being linked to an account-header record.
d) a directory of account header records is recorded on each pack so as to guard against failure of the disk drive or pack usually identified as the VTOC pack; this is the only shared data from pack to pack.
e) the account directories on each pack are divided into two classes; the "accounts on this pack" and the "account not on this pack" classes. An account header record further contain the "pack no" (data base serial number) where its file data reside.
f) account header records are are maintained in sorted order for reporting, but are accessible randomly ("hash-coding" on account name) for rapid retrieval.
g) file header records are maintained in sorted order for reporting, but are accessible randomly (hash-coding on two keys - account name and file name).
h) account directory and file directories are segregated on a page range in the disk, separate from the file data records.
i) file data records are compacted and stored on a page basis, under control of a bit map for rapid storage; retrieval of file data is via chains; furthermore, file storage is under control of a "minimum number of cylinders" policy.
j) statistics are constantly maintained in the data base for pack activity I/O counts), pack storage (how much space left); histograms are maintained on file, last activity (get, update) ageing; the combination of activity, storage and age parameters permits positive control on load equalization and the size of data base to be maintained.
k) most important, the first level of software above the DMS data base manager routines is designed to be completely self-contained so as to detect any errors, recover from malfunctions (incomplete updates due to crashes are reflected by broken chain/wrong bit maps) and continue operation; i.e., catastrofic errors would not propogate across files on a pack. Furthermore, positive controls are given to the operations personnel so as to be able to direct data base repair activities, without exhaustive searches.

2.3 Structure Diagram and DDL

In this section, the ARCHIVE base data structure diagram (Figure 5) and the DDL (Figure 6) are included for reference purposes. Note the simplicity of the data base and yet it exercises a large subset of the data structuring, storage and retrieval capabilities of DMS.

2.3.1 Operations Services Overview

Several utilities have been designed to effect a large degree of control over the data base. A processor was designed to age files from public to private storage; its parameters are ageing dates (how old a file must be so as to be moved into ARCHIVE), data base serial numbers, exception accounts, inclusion accounts; this utility is run every day on a routine basis and at any other time when public storage is down to a minimum. A utility processor was written to logically follow through every acchdr, filehdr, and data pages, with detailed logical consistency checks performed, packmap reconstructing, statistics gathering (distribution of ages, sizes) page re-initializing, and exhaustive directory listing. A utility processor was designed to migrate files to tape with ageing parameters, account and pack parameters. In addition to these various tools provided to the operations staff, the regular DMS- DUMP utility is used daily for a full ARCHIVE save.

2.3.2 Statistics on Data Base Activity

The ARCHIVE system has been fully operational on the two RTCC Sigma 7's for 12 months.

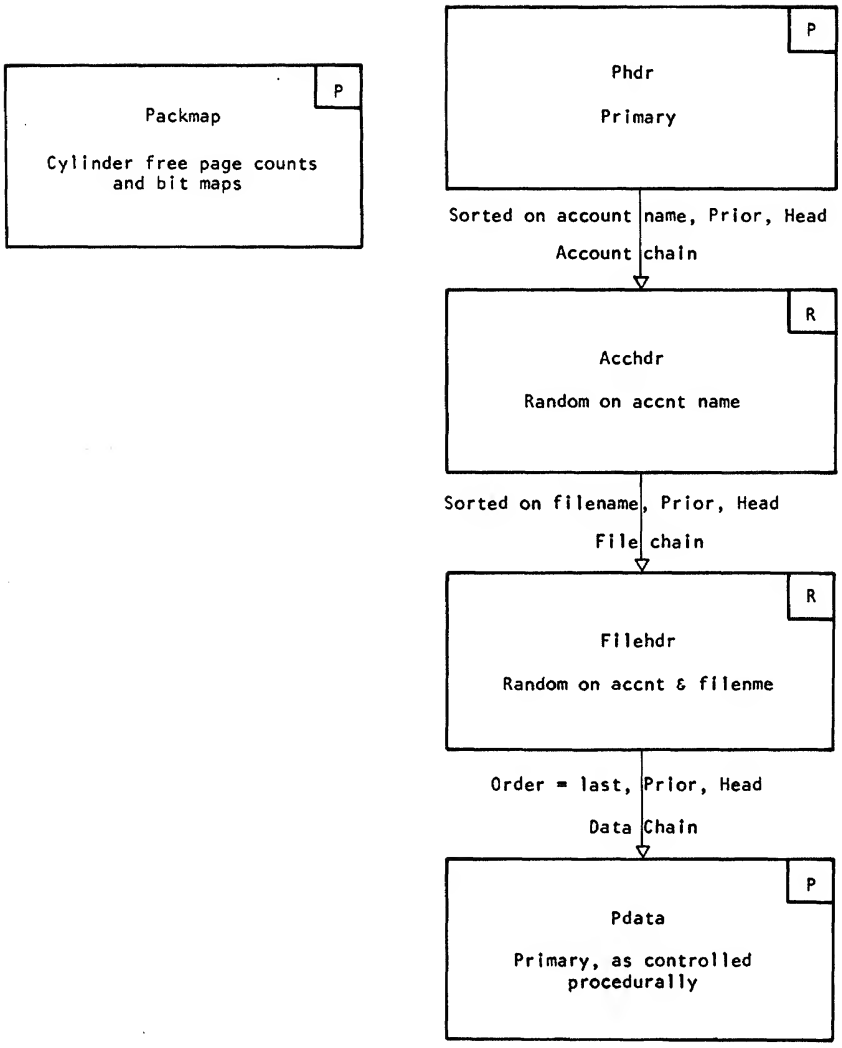


FIGURE 5 ARCHIVE DATA BASE DATA STRUCTURE

```
SCHEMA NAME IS ARCHSCHEMA
    ; PASSWORD IS 'PAL

AREA NAME IS ARCHIVE CONTAINS 118800 PAGES
    ; NUMBER IS 1
    ; CHECKSLM IS REQUIRED
    ; JOURNAL IS NOT REQUIRED
    ; ENCIPHERING IS NOT REQUIRED.
GROUP NAME IS ACCHDR
    ; NUMBER IS 100
    ; WITHIN ARCHIVE, RANGE IS PAGE 1 THRU PAGE 2
    ; LOCATION MODE IS DIRECT
    ; STATISTICS ARE NOT REQUIRED.
  PACKNO; TYPE IS BINARY.
  INUSE; TYPE IS BINARY.
  INV; TYPE IS BINARY.
  PVKNT; TYPE IS BINARY.
  LASTSN; TYPE IS CHARACTER,4.
  REFCODES; TYPE IS BINARY
    ; OCCURS 2 TIMES.
  NOPACKS; TYPE IS BINARY.
  ACCKNT; TYPE IS BINARY.
  TFILEKNT; TYPE IS BINARY.
GROUP NAME IS ACCNT
    ; NUMBER IS 200
    ; WITHIN ARCHIVE, RANGE IS PAGE 3 THRU PAGE 990
    ; LOCATION MODE IS CALC USING ACCNO DUPLICATES ARE NOT ALLOWED
    ; STATISTICS ARE NOT REQUIRED.
  ACCNO; TYPE IS CHARACTER, 8.
  ACCNTLOC; TYPE IS BINARY.
  PAGELIM; TYPE IS BINARY.
  PAGEKNT; TYPE IS BINARY.
  INKNT; TYPE IS BINARY
  FILEKNT; TYPE IS BINARY.
  AGESTATS; TYPE IS BINARY
    ; OCCURS 4 TIMES.
GROUP NAME IS FILEID
    ; NUMBER IS 300
    ; WITHIN ARCHIVE, RANGE IS PAGE 3 THRU PAGE 990
    ; LOCATION MODE IS CALC USING FILENME ACCNOF DUPLICATES ARE NOT ALLOWED
    ; STATISTICS ARE NOT REQUIRED.
  ACCNOF; TYPE IS CHARACTER, 8.
  FILENME; TYPE IS CHARACTER, 16.
  BTMTYPE; TYPE IS BINARY.
  SIZE; TYPE IS BINARY.
  FSTATS; TYPE IS BINARY.
  DATE; TYPE IS CHARACTER, 16.
  FPASSW; TYPE IS JHARACTER, 8.
  BACKUPF; TYPE IS BINARY.
  TAPENO; TYPE IS BINARY.
  ARLOC; TYPE IS BINARY.
  ARSIZE; TYPE IS BINARY.
  FLAGS; TYPE IS BINARY.
GROUP NAME IS PACKMAP
    ; NUMBER IS 500
    ; WITHIN ARCHIVE, RANGE IS PAGE 1 THRU PAGE 10
    ; LOCATION MODE IS DIRECT
    ; STATISTICS ARE NOT REQUIRED.
  GRMAP; TYPE IS BINARY
    ; OCCURS 400 TIMES.
```

Figure 6 - ARCHIVE Data Base DDL

```
  FREEGR; TYPE IS BINARY
    ; OCCURS 100 TIMES.
  NOTES; TYPE IS BINARY
    ; OCCURS 4 TIMES.
GROUP NAME IS FDATA1
    ; NUMBER IS 700
    ; WITHIN ARCHIVE, RANGE IS PAGE 991 THRU PAGE 9900
    ; LOCATION MODE IS DIRECT
    ; STATISTICS ARE NOT REQUIRED.
  ACCNO; TYPE IS CHARACTER, 8.
  FILENME; TYPE IS CHARACTER, 16.
  BTMTYPE; TYPE IS BINARY.
  SIZE; TYPE IS BINARY.
  FSTATS; TYPE IS BINARY.
  DATE; TYPE IS CHARACTER, 16.
  FPASSW; TYPE IS CHARACTER, 8.
  BACKUPF; TYPE IS BINARY.
  TAPENO; TYPE IS BINARY.
  BLOCKNO; TYPE IS BINARY.
  FDATA; TYPE IS BINARY
    ; OCCURS 486 TIMES.
SET NAME IS ACCCHN
    ; OWNER IS ACCHDR
    ; ORDER IS SORTED
    ; LINKED TO PRIOR
    ; STATISTICS ARE NOT REQUIRED.
  MEMBER IS ACCNT
        ; INCLUSION IS AUTOMATIC
        ; LINKED TO OWNER
        ; SET OCCURRENCE SELECTION IS THRU CURRENT OF SET
        ; ASCENDING KEY IS ACCNO
            DUPLICATES ARE NOT ALLOWED.
SET NAME IS FILECHN
    ; OWNER IS ACCNT
    ; ORDER IS SORTED
    ; LINKED TO PRIOR
    ; STATISTICS ARE NOT REQUIRED.
  MEMBER IS FILEID
        ; INCLUSION IS AUTOMATIC
        ; LINKED TO OWNER
        ; SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER
        ; ASCENDING KEY IS FILENME
            DUPLICATES ARE NOT ALLOWED.
SET NAME IS DATACHN
    ; OWNER IS FILEID
    ; ORDER IS LAST
    ; LINKED TO PRIOR
    ; STATISTICS ARE NOT REQUIRED.
  MEMBER IS FDATA1
        ; INCLUSION IS AUTOMATIC
        ; LINKED TO OWNER
        ; SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

END.
```

(Figure 6 Cont'd)

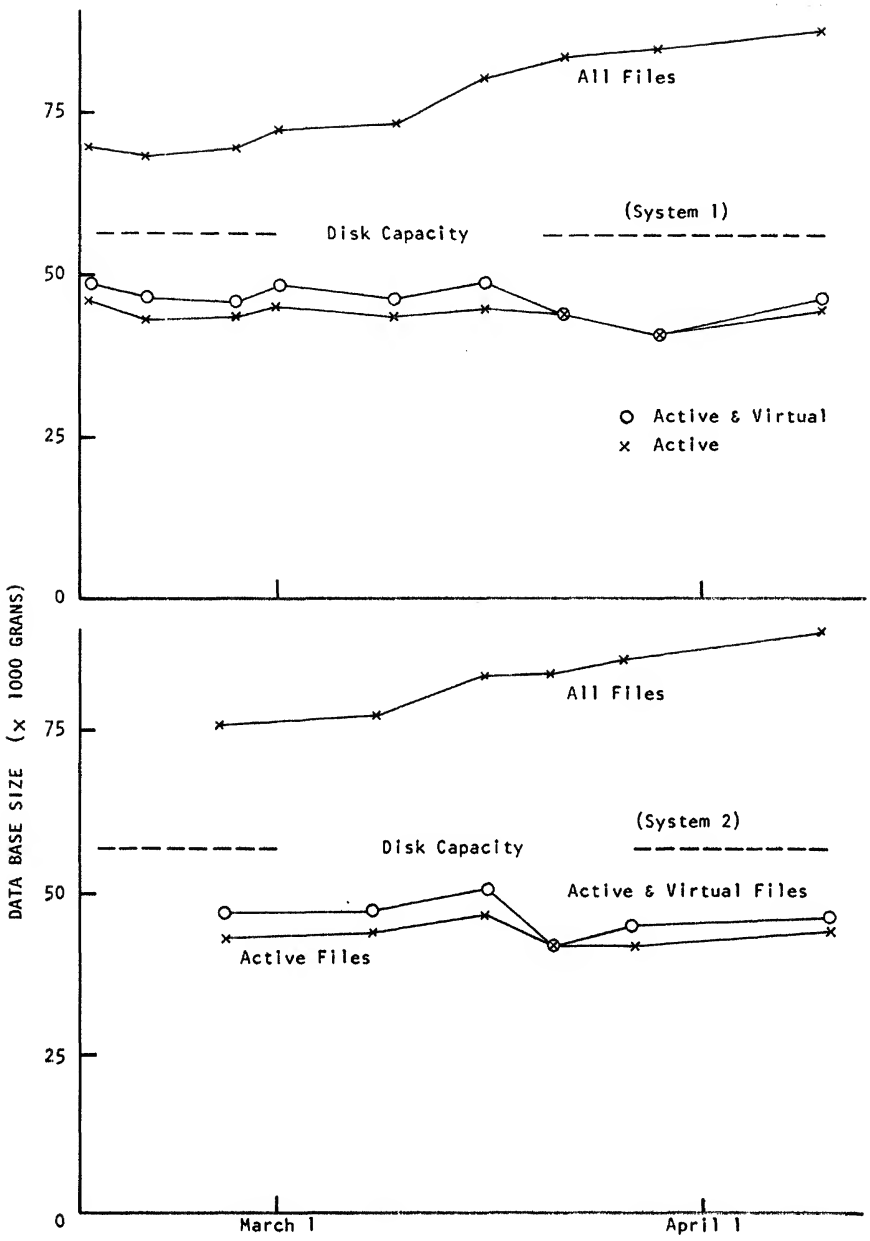

All Files
75
(System 1)
Disk Capacity
50
O Active & Virtual
× Active
25
0
All Files
75
(System 2)
Disk Capacity
Active & Virtual Files
50
Active Files
25
0
March 1
April 1
DATA BASE SIZE (x 1000 GRANS)

FIGURE 7

The growth of the data base in the past two months is shown in Figure 7. As can be seen, approximately half the user files have migrated to tape. During a typical day, there are 1000 update activities against the ARCHIVE packs (per system), the great majority in the saving of new or updated files. Forty-three files on average are retrieved per day per system from ARCHIVE migration tapes causing TARCH (the Tape ARCHIVE Retrieval Program) to be scheduled eight times and requesting one or two tapes per session (1000 request/month/system). The typical user request is for a file or two, with occasional clusters of ten files from a given tape.

The statistics show a steady use pattern distorted by a few identifiable events such as machine failures or single users updating massive number of files (100-200) as in some library constructions or copying a set of files from a user tape into ARCHIVE.

As we said in concluding the first ARCHIVE report (Crean (1972)), we can continue to report that we have found great benefits in the ARCHIVE multi-level storage system and, to date, no user files have been lost from the ARCHIVE data base.

3. TECH REP DISPATCHING APPLICATION

3.1 This application is briefly discussed to examplify a situation of a data base containing two areas, each area having distinctly different functions. The first area contains a large number of records (Machine records) which can be accessed directly by machine serial number or through a geographic/team directory, this area is "relatively static"; i.e., the number of update/delete transactions per day is relatively small with respect to the total number of "Machine Records". The second area contains a highly dynamic, queue driven Service Call Activity Record population. We first describe the problem context, data base structure, and a few key transactions.

3.2 Clarification of Problem

Currently, Tech Reps, for service-oriented organizations, are dispatched to a customer site by identifying customers on a microfiche branch service directory and by manually preparing a manual routing slip to record the service call. This method is an improvement over the previously used card files but the following problems are still apparent:

1. Delay in retrieving customer's record. When a customer calls for service there is some delay in retrieval of the customer's record, particularly if it is not on the microfiche.
2. Dispatcher manually completes a Routing Slip. There is an additional delay as the operator writes up the Dispatch Routing Slip and sends it on. Data entered - Serial Number, Tech Rep Assigned, Time Call Received, Nature of the Problem, and Customer Name.
3. Manpower Utilization. If too many calls are received for one Tech Rep, the calls are not usually dispatched to another Tech Rep with a much lighter load.
4. Identification of Problem customers sooner. At the time a customer calls, the Dispatchers have no history information immediately available.
5. No fast retrieval to re-dispatch. If calls must be re-dispatched to a Tech Rep, the correct Routing Slip must be selected. This is difficult and time consuming due to the number of slips to be searched.
6. Call Completion. When the Tech Rep calls in that he completed a call, the correct Routing Slip must be retrieved and the data manually entered on the document.

3.3 Terminal oriented, structured data base solution. With an interactive terminal-oriented system, the following goals can be achieved:

1. Easier retrieval of customer records. Because retrieval can be speeded up, more customer calls can be processed. Further, few customers would receive busy signals since the telephone lines would be freed sooner.
2. Entry for a new machine's record could be accomplished in minutes, for a Tech Rep need only phone the dispatch office to state that a machine has been installed.
3. Eliminate manual Routing Slips. Since the customer information is already in the computer file, the operator would enter a call into the computer by keying in only a problem code.
4. Manpower Utilization. As the call load on a particular Tech Rep builds up, a report can be displayed on a CRT to a clerk monitoring the Tech Reps. At the same time, adjacent Tech Reps (from the same team) who have light loads at that point can be displayed.
5. Identify problem customers immediately. Since the computer file contains service history for each customer call, it can quickly determine its service history.
6. Identify calls during the day that are not meeting required response time parameters in order to take immediate corrective action.

3.4 Data Structure Diagram

Figure 8 shows the data structure diagram. The left-hand side of the page (from the middle of the page to the left edge) depicts the structure for the Machine Record area of the data base. Note the various ways a Machine Record can be retrieved; i.e., (1) directly by Machine serial number through the Machine PTR set; (2) via a Tech Rep record through the Territory set. The second area contains the dynamic Service Call area: a service call record proceeds from queue to queue as it gets assigned to a Tech Rep, reported upon and completed.

3.5 Services Provided

This section briefly outlines the terminal oriented services which were provided by this system.

Customer Operators:
- Machine lookup, including service history
- Service call status checks
- Correction of errors in customer records
- Service call entry or cancellation

Dispatchers:
- Displays for radio dispatching
- Displays for phone-in dispatching

Tech Rep Operators:
- Display of a tech rep's calls
- Call load summary for a team of tech reps
- Transfer of a service call from one tech rep to another
- Logging of completed service calls

Branch Managers:
- On-line inquiries into customer and service call records
- Snapshot of service call activity for team or branch
- Reports of critical calls
- Daily summaries of service call activity, calls left at end of day, possible problem accounts,...

3.6 Closing Remarks on the Tech Rep Dispatching System

The system was first implemented as a pilot program and took 8 man-months of design, coding and debugging for a complete transaction processing system, with content switching, backup and recovery. DMS was absolutely critical in this application which required elaborate data structuring. Almost all of the facilities of DMS were utilized in this application. Checkpointing the data base was crucial and reduced the incidence of errors from an intolerable level to a barely perceptible minor annoyance. This application could not have been

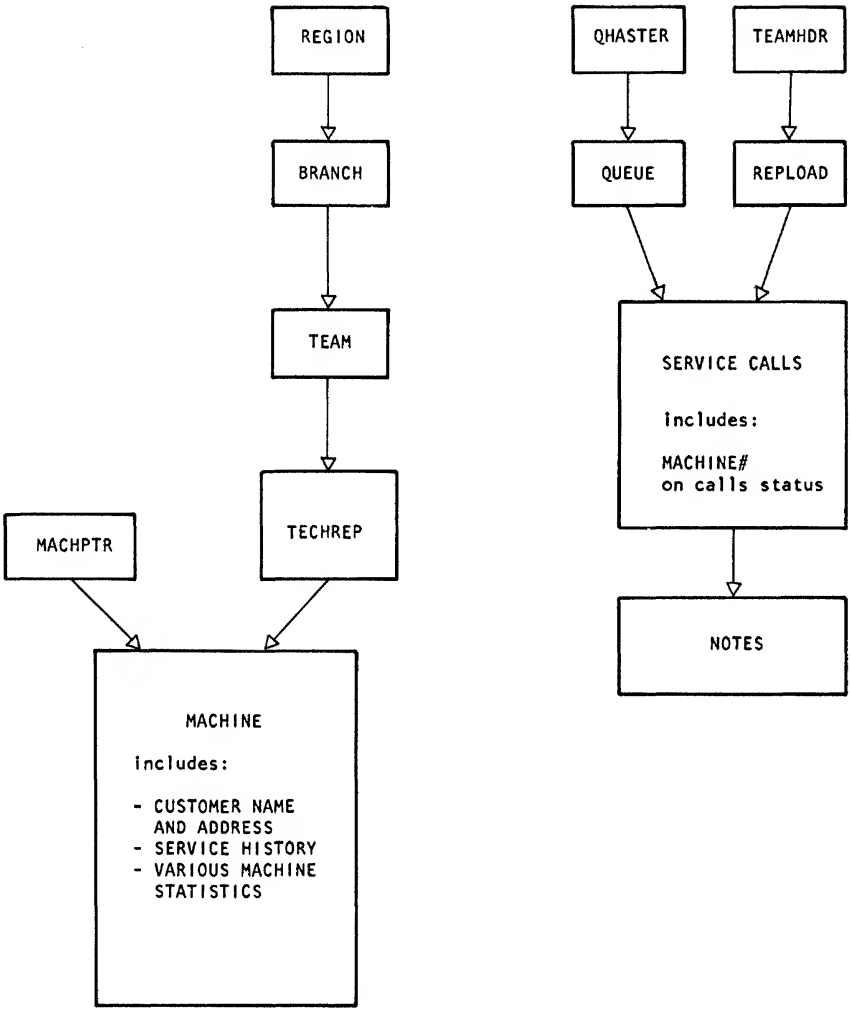


FIGURE 8 TECH REP DISPATCHING DATA BASE

accomplished in so short a time and at so comparatively modest cost without the structured data base capabilities offered to us by DMS.

4. IDDP APPLICATION: An Interactive DMS Data Base Editor

4.1 Introduction

In the light of a growing number of DMS applications, it was found that an Editor for DMS was required, that would be able to interactively access any DMS like data base and perform any DBM action; i.e., interactively exercise any DML verb. For maintenance, tutorial, record breakout, record set printout and many unplanned activities an interactive, data base independent editor was found to be invaluable for data base administrators. IDDP stands for Interactive DMS Data Base Debug Package (Ohayon (1972)). The IDDP system has been designed to fully exploit the capabilities offered by DMS, the procedural Data Management System developed by XDS (Xerox (1972)), in a non-procedural manner.

The system provides a language that allows the user to interact with the data base by letting him invoke the DBM procedures using the area names, group names, item names, and set names as defined in the Data Definition Language (DDL) of the data base. However, the language does not provide the capabilities for asking questions of any logical complexity.

IDDP has been designed to provide extensive tutorial information about the user data base, automatic input and output formatting, and dump facilities. The most important feature of IDDP and (and probably the most interesting) is that IDDP is truly data base independent. IDDP interfaces with any DMS data base through a file known as the Query Schema file.

Like any Editor, IDDP is a self-contained system in which the user specified the information he requires while the process by which it is obtained is built into the system. This process assumes the existence of a file known as the Query Schema file, built by a special processor the Query Schema Processor. The Query Schema file is the only interface needed by IDDP to access a particular DMS file. Therefore, IDDP achieves Data Base independence through the use of the Query Schema file.

IDDP is written mainly in FORTRAN and is made up of a main program and a set of 70 commands executable interactively at the user's request. IDDP is not designed to be called by object programs, but to be used independently of them.

4.2 IDDP Operational Interface

4.2.1 Query Schema and Schema

The Data Base Manager (DBM) interfaces to any DMS data base through the Subschema file. IDDP interfaces the user to DBM through a file known as the Query Schema file. The Query Schema is an alphanumeric keyed file relating the names defined in the DDL to their numeric offset (i.e., working storage increment) recorded in the Subschema. This correspondence between numeric offset and DDL names allows IDDP to interpret a user's request expressed in terms of DDL names and to issue a DBM call with appropriate offset values.

4.2.2 The Query Schema Processor (QSP)

The Query Schema file (QSF) is generated by a processor called the Query Schema Processor or QSP.

The QSP records into the QSF all DDL names with their subschema addresses. At run time, IDDP uses the subschema address of a given name to extract from the subschema the characteristics of this name.

The QSP produces optionally:

1. The Query Schema file
2. A listing of the DDL names with their working storage offset, and subschema pointer
3. A topology table describing the relationship between groups and sets

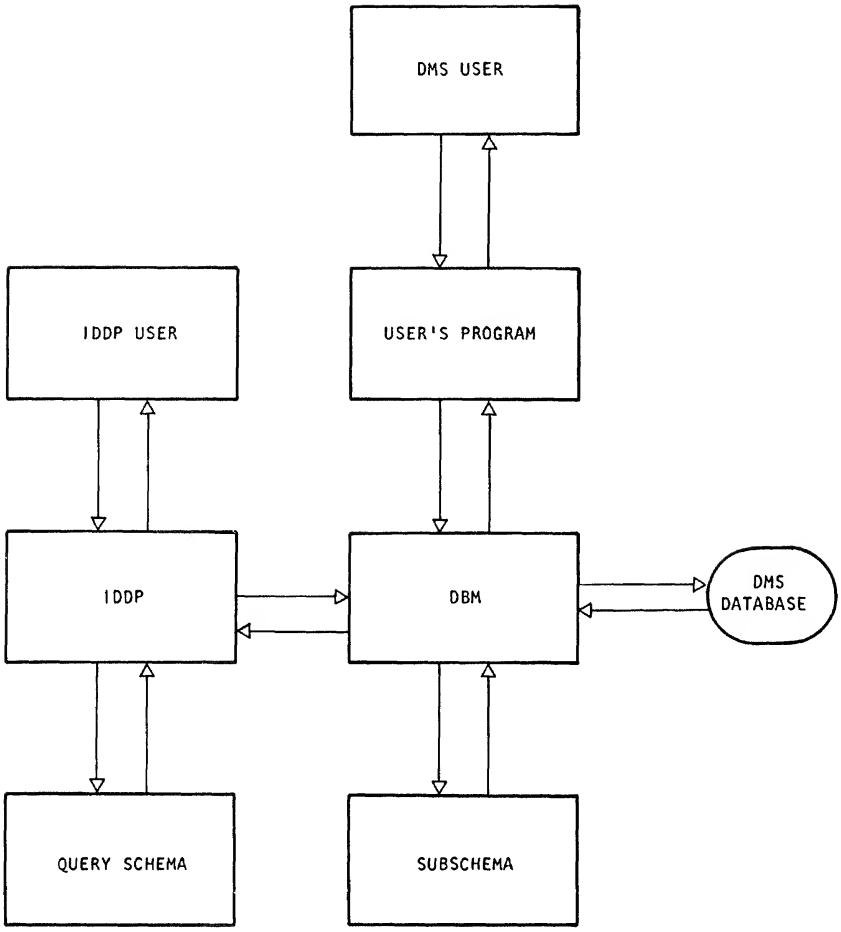


FIGURE 9 IDDP USER AND DMS USER

Figure 9 shows that with IDDP no other program is necessary to access a DMS data base as long as the Query Schema file is provided.

4. A sorted listing by DDL names

4.3 IDDP Commands Overview

IDDP allows a user to call any DMS procedure (i.e., any of the DBM routines) with an IDDP command with same name as the DMS procedure. These commands are called the DBM commands. In addition, commands were built into IDDP to assist the user in the execution of the DBM commands. These commands are called I/O control commands and they provide a way to:

a) Control the amount of input to be requested from the user before the execution of the DBM commands and the input format.
b) Control the amount of information to be printed after the execution of each DBM command and the output format.
c) Control where the output is to be printed.

Arguments are never required with command names. If they are needed, but not supplied with the command, IDDP prompts for their values. When IDDP prompts for argument values, the user types one line at the terminal followed by a carriage return. If the first character typed is a carriage return, the user has typed a 'NULL ANSWER' which (when acceptable) signifies to IDDP to use the previous argument value or a default value. As an example, let us consider the MODIFY command. The following three syntax forms are acceptable:

a) MODIFY
b) MODIFY group-name
c) MODIFY group-name item-name

In addition, when IDDP expects any argument value it always indicates specifically what is expected and how it is expected. If the argument value does not correspond to what is expected, an appropriate message is typed at the user console and IDDP reprompts for the argument value. For example, if the first form of the MODIFY command is used, IDDP starts by prompting for a group name. If the user answers by giving the name of an item instead of the name of a group IDDP reprompts with 'WRONG GROUPNAME.GROUP NAME='

4.4 IDDP Concluding Remarks

IDDP has proven to be a widely accepted and powerful tool which has been used in all phases of projects involving DMS data bases. IDDP has been used in data base design and development, in data collection, in editing and maintenance activities. Its data base independence characteristics through usage of the Schema and Query Schema files have made it extremely acceptable to a large audience.

5. CONCLUDING REMARKS

All of the advantages that were touted about integrated data base systems via DBTG-like Data Management Systems were found to be indeed true. Once the initial learning phases have been completed for applications groups and after several persons have enough knowledge to assume a Data Base Administrator role, the benefits of Structured Data Bases come about. Major criticisms have come from applications areas where "flat" files were converted to structured files for the convenience of minor users at the sacrifice of major users for which "flat" file accessing and elaborate report generators were highly developed. This leads us into comments about drawbacks of DBTG-like systems. Since these systems are in their early phases of development and acceptance, they are surrounded by barriers which make them "hard" to use. The first major enhancement to DBTG-like systems, must be the specification and implementation of a Report Generation Facility over highly structured data bases (Batch or Terminal-oriented). This in itself would greatly increase the user population. Secondly, interface specifications to other compilers (FORTRAN, PL/1, COBOL, etc.) must be developed; even more important, interactive compiler interfaces (APL, BASIC) through Query Schemas (Ohayon (1972)) would make access to DBTG-like systems far

easier than through COBOL. A data base administration language which is data base independent and permits repair, editing, and maintaining of the data base, must become specified and standardized (Ohayon (1972)). Finally, various kinds of easy to use query languages, restricted in scope and question structure must be specified and implemented (for example, see Lavallee (1972)) so as to be able to extract data and short reports from structured data bases, without the need to COBOL-ize user demands for data. All these recommendations are being studied in various degrees of depth and it is hoped that the next few years will see a good measure of activity and results in these areas.

DISCUSSION

SHEEHAN:

DBMS systems that I have seen have not provided any help to the user in trying to load the file and keep data integrity. Is it true that at the time you define the file you can say that data in a given file section, segment or whatever, must conform to the same set of rules?

OHAYON:

Yes, that is correct. You can do that on range checking as well as data format at execute time.

SHEEHAN:

Are there user exits available at data base definition time which allow you to say: 'I want this much editing done'?

OHAYON:

When you try to store some information and the information does not correspond to the data definition, control is return to you.

SHEEHAN:

Are there exits that you can add so that there would be additional checking done for you both at definition time and later?

OHAYON:

In the data definition language, you cannot say that you want to execute a certain action if you have this type of error. It is when you write your application, that you let the system return. The editing that you are doing is completely within the confines of the field.

JARDINE:

At least one system, the Burroughs 6700 Data Management System, allows user exits in the data definition language. It permits the definition in the DDL of the name of the procedure to be invoked dynamically at execution time either before or after the execution of any of the data manipulation statements. Therefore, I can specify: before Delete, execute some system defined procedure. This procedure has addressability to at least the same data as the program that is attempting to do the Delete. For editing and validation, this is a valuable tool.

WIEDERHOLD:

Do you provide any option whether to invoke the editing or not? We have automatic editing in our system and we found that when data entry was done by secretaries the editing was self defeating. They couldn't handle some of the problems. We now have the option to defer the editing.

OHAYON:

No. When you decide that you want the editing, the editing will take place automatically.

Data Base Management Systems, D. A. Jardine (ed.).

DATA BASE FACILITIES FOR THE END-USER: PRESENT AND FUTURE

P. L. NICHOLS
Ontario Hydro
Toronto, Ontario

INTRODUCTION

Most of the effort within the data processing community has been directed at satisfying the needs of our own professionals, be they application programmers, system designers, or Data Base Administrators. While these needs are extremely important, it is now essential that we turn our attention to the needs of the end-user, whatever his profession or area of work within his organization might be. Only by bringing the capabilities of the computer and data base techniques directly to the user, will we be able to fully exploit the potential of computerized data in the years ahead.

The user who is the subject of this paper, is a person who will not be required to have any special knowledge of computers or programming. His job may be associated with any job discipline (e.g. engineer, lawyer, clerk, scientist, manager); he may work in any functional area within his organization (e.g. engineering department, records centre, research laboratory); and he may be at any level within that organization, from clerk to president.

The prime tool for providing the capabilities of the computer to the user will be a User Language. This paper will describe some of the deficiencies of current User Languages*, and will define requirements which future products should satisfy.

A User Language will provide the user with great flexibility, will eliminate the need for him to define all of his requirements in advance, and will provide him with a direct interface to his data, thus reducing the time lag between question and answer. By enabling a user to meet his own needs as he sees them, we will amplify his effectiveness in dealing with new and constantly changing situations.

In providing the user with a direct interface to his data, we must accept the fact that the data within an organization is the property of that organization and its users; it does not belong to the data processing department or the Data Base Administrator. We must protect the data base, but we must not prevent the user from exercising his responsibility for the data. Today, however, this principle is violated in many companies and the data processing department controls the corporation's computerized data, and determines who can use that data. In most cases this situation began many years ago when computers were first introduced to a company, and users were afraid of this new technology and content to let "the experts" have their way. Over the years these users have become more knowledgeable about the capabilities of computers, and at the same time have become somewhat disillusioned with the results of their data processing experts.

It is now essential that we provide these users with a tool which will permit them to make more productive use of computerized data in carrying out their jobs in an effective and timely manner; this tool must allow them to use data directly without having to go through an intermediary in the data processing department. This tool is a User Language.

*Comments on deficiencies of current products are based on the author's experience at Ontario Hydro with the following Data Base Management Systems and User Languages: SYSTEM 2000 (MRI), IMS2/IQF/GIS (IBM), MARK IV (Informatics), TOTAL (Cincom), and DMS-1100 (UNIVAC).

THE NEED FOR A USER LANGUAGE

The data processing industry has now reached the point in its evolution where it realizes that is simply not possible for a user to determine in advance all of his information needs. Much information is needed quickly, on a one-time basis, to solve a particular problem. Different users have differing needs for information, depending on their own work style and the changing situation from day to day. As a result, a flexible approach is needed, where the user is given a direct and independent interface to his data. The traditional data processing approach fails to meet this need. For users, the process of explaining their requirements to a programmer, waiting (and paying) for it to be programmed, documented, and tested, is frustrating and agonizingly lengthy. Indeed with this approach, it may take the user several days to get his answer (by which time the information may be totally useless), and he may never have the same requirement again. The only answer lies in providing the user with a User Language, which he can use to solve his own problems, when they arise, without having to depend on someone else.

SCOPE OF A USER LANGUAGE

Many of the User Languages in existence today have been designed as adjuncts to a Data Base Management System, almost as if they were an afterthought. In fact in some cases, such as TOTAL and DMS-11000, no User Language is provided at all. One must conclude the vendors in general believe that problems of any significance or complexity will be solved by application programmers, and that users are only capable of dealing directly with the computer in trivial matters. A fundamental change in this attitude is required. Users are capable of handling much more complex problems that we give them credit for; after all they managed quite well before we computer specialists came along. But in order for them to use the computer actively in solving their day-to-day problems, we must provide them with the proper tool. Today's products fall far short of being that tool. Presumably developed on the basis that they will only be used to carry out trivial functions, they simply do not provide sufficient capabilities to allow a user to perform his work effectively by using these tools.

RELATIONSHIP TO DATA BASE ENVIRONMENT

Although the User Language is the prime tool for the user, it will depend upon the existence of several other components within the framework of a Data Management System. In particular it will depend upon the following:

1. A data base
2. A Data Dictionary, which will contain the descriptive information about the data in the data base
3. A Data Base Administrator (DBA), responsible for various control functions, and for providing consultation and help to users
4. A Data Base Management System (DBMS), the basic software component of the system, through which all input/output operations on the data base will be performed
5. A Data Communications Function, providing terminal transparency and the interface to all terminal oriented operations.

The component of the Data Management System which processes the User Language will be referred to as the User Language System or ULS. In discussing a User Language it is also necessary to consider the ULS, for the two will appear as a single entity to the user. In an implementation, the ULS will probably be a combination of hardware and software, and will interface with the other components listed above.

EASE OF USE

The User Language must be simple, easy to use, and easy to comprehend; it should be so easy to use that it does not distract the user from the problem he is trying to solve. It is an overriding requirement that the User Language optimize the human resource, probably at the expense of machine resources (which are becoming cheaper all the time anyway).

The User Language must at least appear to a given user as a single language, and he should not be required to use different languages in different environments. This applies at three levels:

1. the language must support either batch or interactive execution (it is assumed that preparation and submission of User Language requests will be done interactively; the need for batch execution will probably remain for some time however). The user himself will not think in these terms; he will prepare his request interactively, and will then select an option for execution in terms of response time and cost. The User Language System should present the user with a series of alternatives from which to choose; batch execution will be equivalent to a long response time.
2. the user should be able to use the same language regardless of the particular Data Base Management System employed at his installation.
3. the language must be independent of the computer hardware used in process it.

The computer industry recognized the need for a single standardized programming language, independent of computer hardware, a decade ago, and subsequently evolved an industry standard - COBOL. The need is obviously even greater for a standard User Language, and will be essential if users are to use computers actively. The thought of having to learn two or three different languages will be enough to deter even the most zealous of users.

SYSTEM 2000 provides a User Language which can be used on three different vendor's computer hardware, a notable achievement with today's technology; in order to use that User Language, however, the installation must also use the other components of the SYSTEM 2000 Data Base Management System. No vendor today provides a User Language which can be used with any or all DBM's; MARK IV provides a good initial step in this direction however, as it has been successfully interfaced with IMS and TOTAL data bases. With regard to item 1) above, only SYSTEM 2000 provides a single language which can be used in either batch or interactive mode. In the case of an IMS data base, the user must use two different languages if he needs to operate in both modes; this poses a severe training problem for the user.

Minimal training should be required to use the User Language, and various training aids in a variety of forms should be provided by the vendor. Such forms might be audio-visual, learner-controlled-instruction, computer assisted instruction, or class room courses. The training courses should be modular, with at least a basic and advanced stage; this permits the user to initially develop confidence in his ability to use the language, without overwhelming him with all its capabilities. The user's initial reaction is extremely important to his future attitude and success in using the language.

The User Language System must contain a comprehensive prompting capability, to assist the user in preparing and entering his requests. More experienced users who are thoroughly familiar with the system should be able to inhibit this prompting. The ULS should be forgiving. If the user has made a mistake, it should be easy to correct. Related to this is the need for the messages and responses provided by the ULS to be positive, helpful, and easy to understand. Whether the responses are error messages, information diagnostics, or simply part of the dialogue with the user, they should be formatted and displayed in understandable user language. The reader may conclude that this requirement is obvious, and so it should be, but unfortunately many messages provided by current User Languages can hardly be understood by a programmer, never mind an unsophisticated user.

COMMUNICATING IN THE USER'S TERMS

From the user's point of view, use of the User Language will probably be an incidental part of his job. Thus the user will regard the User Language as just another aspect of his normal work routine, and his only contact with the computer will be via the preparation and submission of User Language requests. This means that the User Language should be as close to the natural language of the user as possible. The goal in developing the User Language should be to require the user to adapt as little as possible from his normal mode of communication.

Each class of user operates within a certain "world" - that world being distinguished by a specific job discipline and functional area of work. Each world, and therefore each class of user, has its own unique vocabularly and mode of expression, i.e. its own "natural" language. There is a common core of language that is used by all people no matter what their job discipline and area of work. Each job discipline, however, also has its own unique "jargon", which in general cannot be understood by someone outside that world. We all realize for example that without medical training we can't understand the doctor's technical diagnosis, and without training in geology we can't make much sense of the geological description of the moon. The Venn diagram, Figure 1, taken from Plagman (1972a) portrays this situation graphically.

Conceptually, the User Language can be considered as two distinct components, although from the user's point of view it will be a single language, and in fact only one of the components will be used by certain classes of users:

1. that part of the language which is common to all people; this will be "natural language" in the normal sense of the term. While it is recognized that the use of pure natural language is not within the present state-of-the-art, it is now technically feasible to support a natural-like language.
2. that part of the language which is unique to those in a particular job discipline and functional area of work. I have used the term "world" to describe this environment which is unique to a given class of users, and in essence it defines the context within which the terms, phrases, and symbols used by the user are interpreted.

Within some worlds, such as the typical business environment, the second component described above will also be natural language. In other worlds, however, there are no words at all, merely numbers, quantities and symbols, stated within a highly rigid structure. For example, in areas such as drafting and engineering, the User Language may consist solely of the second component, formed from the standard symbols of the trade.

In developing the User Language, the vendors must recognize and support these different "worlds"; specifically the Data Base Administrator must be able to adapt the language to the specific needs of his organization. Modules supporting certain worlds should be delivered by the vendor as part of the User Language System; the DBA should be able to select and/or tailor these according to his users' requirements. The vendor should also provide a facility whereby the DBA can develop and install his own modules; these may be required to support particular job disciplines or a different national language (e.g. French, German), not supported by the vendor.

The User Language should impose minimal constraints on the user in naming data. Each user should be able to use his own nomenclature to reference data - he must not be forced to use a standard name which may be meaningless to him; this can be achieved by allowing the user to assign aliases. Every piece of data will also have a unique official name, however, for use by the DBA in co-ordinating and controlling the multitude of aliases that will arise.

The end-user must have significant latitude in the way he expresses data values. For example, 1, 1.00, 0.1 x 101 could all be interpreted the same by the User Language System. The latitude in data value expression will be dependent upon the definition of the data as contained in the Data Dictionary.

The user must have the ability to refer to data on a time basis for those items so designated in the Data Dictionary. The time basis may be absolute as in "sales for April 1973" or it may be relative as in "John Smith's previous salary".

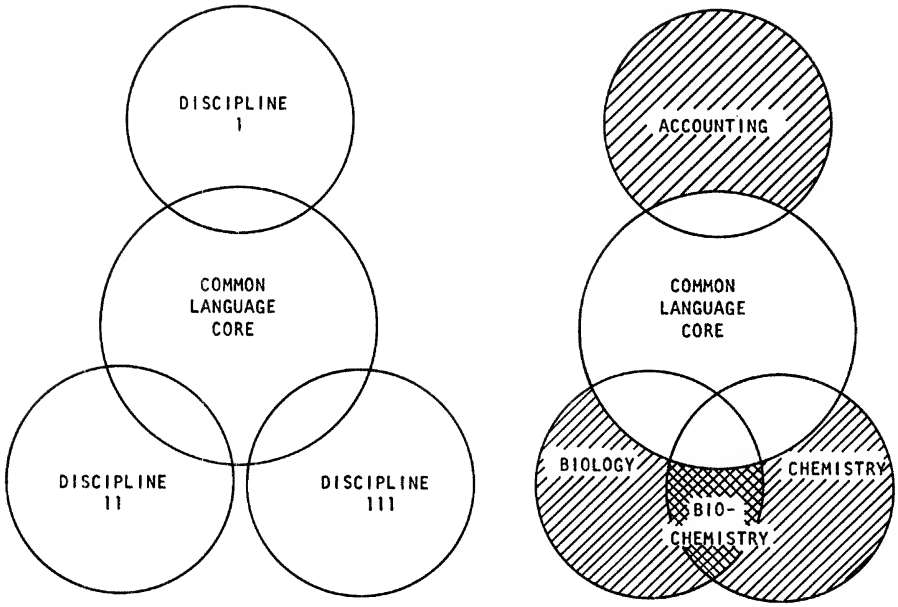

DISCIPLINE
1
COMMON
LANGUAGE
CORE
DISCIPLINE
11
DISCIPLINE
111
ACCOUNTING
COMMON
LANGUAGE
CORE
BIOLOGY
CHEMISTRY
BIO-
CHEMISTRY

FIGURE 1

The data will have a single name, such as "sales" or "salary" in the examples above, but it may have a time modifier.

The User Language will thus be much closer to the native mode of communication of the user than any languages which exist today, although there may still exist certain rules and restrictions on sentence structure which the user will have to observe in using the language. In addition, the User Language will be problem oriented; the user will be able to concern himself only with 'WHAT' he wishes to do, not with 'HOW' it is to be done.

Today's User Languages do not come close to meeting these requirements. IBM's Interactive Query Facility (IQF) is a good initial step as it provides a reasonably English-like syntax and a limited capability for the language to be adapted to the world of the user via its "phrase data base"; this latter capability is severely constrained however as there can only be one phrase data base for the entire installation, not one per user (or class of users) which is essential. It, along with all other existing products, does not come close to being a natural-like language, and none provide the ability for an installation to define its own modules to be added to the language.

ABILITY FOR USER TO DETERMINE WHAT DATA EXISTS

One of the most important considerations in permitting users to make effective use of computerized data, is the ability to find out what data exists. The ULS must be able to help the user to

- .determine what data exists relative to his particular function and the specific problem at hand
- .determine the data-name, definition, and other attributes of the data he needs.

This means the ULS must provide the user with the capability to browse the Data Dictionary, which would be treated as a data base for browsing purposes. The user should be able to browse the Data Dictionary on several criteria:

- .world
- .data name, either official name or various user assigned aliases
- .attributes such as size, characteristics (e.g. numeric, alphabetic)
- .data relationships
- .source of the data
- .reports used in
- .authorized users
- .keywords within the narrative definition

When browsing the Data Dictionary, the user should be able to request similarly spelled data-names, or those sounding the same.

A Data Dictionary is required as an integral part of the Data Management System of which the User Language is one component. Ideally it should be used as the source of metadata (information about data) by both the user and the Data Base Management System software, otherwise redundancy of metadata will occur with all the inherent problems that redundant data implies. This concept is described thoroughly in the paper "A Data Dictionary/Directory System within the Context of an Integrated Corporate Data Base", Plagman (1972). No current system of which the author is aware provides such a capability, and as long as this situation exists, an effective User Language capability cannot be achieved.

THE USERS VIEW OF THE DATA BASE

Today's products require the user to know a great deal about the structure and format of the data base, and the form his request takes in dependent upon this structure; this represents one of the most serious problems to be overcome. The provision of an effective Data Dictionary facility, with which the User Language can interface, would go a long way to solving this problem.

The user should not need to know, or be concerned about, what data base(s),

or how many data bases he is using. In addition he must not need to know anything about the physical structure or format of the data base. He must have the ability to address data items (basic, derived, virtual) anywhere in the data base by name. The fact that the data could be anywhere (internal or external storage) should be transparent to the user.

The user must not have to qualify his request to a particular data base; the only level of qualification required will be to a world, and this will be an implicit qualifier. If a user's request is ambiguous, the ULS should prompt the user for qualifying information; the user should never have to qualify beyond the level required to resolve ambiguities.

MODE OF OPERATION

The User Language should support two modes of operation:

1. a mode in which the user is in control, and where he will convey his questions/instructions/etc. to the ULS via User Language statements
2. a mode in which the ULS is in control, and presents the user with either a menu-like list of choices from which to choose, or a series of questions designed to elicit the necessary information from the user.

The user should be able to alternate between these two modes, going back and forth between them depending on the nature of his problem, the difficulties he is encountering, etc.

All of today's products operate in only one mode - that where the user is in control. The technique used in MARK IV of ticking off appropriate boxes is a step towards the second mode of operation, in which the User Language System is in control, presenting the user with a series of questions designed to elicit the necessary information.

THE USER/SYSTEM INTERFACE

The preparation and submission of User Language requests will be done almost entirely in an interactive mode in the years ahead. The user will specify a response time for his answer which will dictate whether exectuion is performed in batch or real-time (see earlier section).

The man/machine interface is extremely important, as the user must first overcome the obstacle of dealing with a physical terminal before he can even begin to use the User Language. Simple mechanisms must exist for signing-on, signing-off, and entering requests; no special control language should be required. The User Language must be adaptable to a broad variety of input/output devices, so a suitable device can be selected for each user. Just as the language must support various "worlds", it must also support various physical environments, ranging all the way from the business office to the machine shop. A given terminal should be adaptable to different classes of users, possibly via the use of templates or interchangeable keyboards.

The ULS should analyze and validate each statement of a request submission as it is entered, and should prompt the user in the event of invalid commands or missing data. Some attempt should be made by the ULS to determine what the user really intended, and to suggest what he might do next.

If the user gets into difficulty he should be able to inform the ULS that he has progressed to a certain point, sign-off, invoke on-line a discussion with support personnel to help him, and then pick up again where he left off. He should also be able to direct his diagnostics directly to the support personnel to enable them to help him more effectively. It is absolutely essential that a user not become so frustrated that he totally abandons his efforts, as this can have far-reaching psychological implications for his future interactions with the system.

The sign-on mechanisms in most current User Languages are anything but simple, and the ability to adapt the language to different terminals is almost

non-existent.

The user should be able to save a request, catalog that request (under a name of his own choosing) for reference at a later time, and execute that request repeatedly whenever needed in the course of his work. When invoking the request at a later time, the user should be able to enter only those portions of the request which are different from the original submission (this would normally be the data fields or values being referenced). This capability, whereby a user can define a request for subsequent use by either himself or a group of users, on a repeated basis, essentially provides the user with the ability to develop an application. When using the request on a recurring basis, users would simply plug in the appropriate data values.

To reduce the need for the user to specify certain information repetitively with each request, and to provide the controls and constraints necessary for an effective operation, a hierarchy of defaults is required within the User Language. Four levels of defaults should be provided:

1. Functional defaults - incorporated in the ULS by the vendor
2. Installation defaults - reflecting corporate rules and standards
3. User defaults defined by the DBA - reflecting defaults unique to a given user or class of users
4. User defaults defined by the User himself - a convenience to the user.

The user must have the capability to override any user oriented defaults by providing other values, provided, of course, that these values are consistent with the installation defined constraints.

Current products generally provide functional defaults, and to a limited extent, installation defaults. The other two levels are generally unavailable.

USER PROFILE

It is a fundamental concept of data base usage that a variety of users with different skills and degrees of authority will need to access different parts of an organization's data bases. It is also obvious that some form of control is needed to protect the integrity of the data base and to prevent unreasonable use of computer resources. That information which describes the characteristics of the user, and defines what he can and cannot do, can be referred to as a User Profile. The information it contains can be grouped into 3 categories:

a) information which restricts the user by virtue of the rules established by the corporation that owns the data base
b) information which is provided to resolve situations which the user is not capable of resolving, e.g. use of system resources or search techniques
c) information which is provided by the user for his own convenience, to aid in ease of use of the ULS.

The User Profile will be established by the DBA, and in most cases it will be maintained and updated by DBA. Information provided by the user for his own convenience (i.e. category c) above) however, will be maintained and updated by the user himself.

The following list gives an idea of the type of things required in a User Profile:

- .the data items a user can retrieve
- .the data items a user can update
- .limits on the machine resources which a user can monopolize
- .priority, maximum cost and account number defaults which can be assigned to a user's request if he doesn't specify different ones
- .the "world" in which a user normally operates (this is used as an implicit quallfier for the data-names used by the user)
- .aliases for data-names and functions which are unique to a given user of class of users
- .the User Language functions which a user may or may not use
- .output devices which a user may use

.the device to which output will be sent if the user doesn't specify one
.the defaults which apply to a given user or class of users, whether defined by the DBA or the end-users themselves.

Present Data Base Management Systems provide the ability to define security, that is to define the data items which a user can retrieve and/or update; none of the other facilities are available.

FUNCTIONAL CAPABILITIES WITHIN THE LANGUAGE

For the purposes of presentation, the functional requirements of the User Language can be divided into 5 areas:

.Data Input
.Data Retrieval
.Data Manipulation
.Data Update
.Data Output

The user, however, would not view the User Language in this way, and the various functions would not be provided to him as distinct functions. The user must be able to state his problem in a way which is natural to him, and any one statement might embody or imply several of the above functions.

Input

The ULS should validate the data values entered as part of a request using the format and validation rules stored in the Data Dictionary. For example, there should be class checks (numeric, alphanumeric, special classes), range checks, specific value checks, and field interrelationship checks.

The user should have the ability to define a transaction format to the ULS which would then be catalogued in the Data Dictionary. This would permit the user to define a fixed preformatted transaction which he could then use repeatedly simply by providing the data values. This facility will enable the User Language to be used for entering large volumes of transactions, if desired.

The User Language should permit the user to enter data from a variety of different terminals, and should prompt the user if he is using an unfamiliar device. Differences between terminal devices should be as transparent to the user as possible.

These capabilities are not provided by current User Languages.

Retrieval

Little need be said about retrieval capabilities as current User Languages are the most advanced in this area, with most of them permitting quite sophisticated searches of the data base on several criteria. In many cases this capability is provided in a way which makes it quite difficult for the user to use however.

Additional capabilities required are:

.tutorial functions for guiding the user through sophisticated searches in a step-by-step learning technique
.the ability to select a subset of the data base by some form of statistical sampling
.the ability to specify the sequence in which the qualified data is to be retrieved and presented to the user.

Data Manipulation

Various data manipulation operators are required for five distinct disciplines:

1. Arithmetic Computations
   e.g. Add, Subtract, Multiply, Divide
2. Business Type Functions
   e.g. Total, Percentage, Mean, Standard Deviation, Maximum, Present Value,

Compound, Amortize, Depreciate
3. Mathematical Functions
e.g. Sine, Tangent, Absolute Value
4. Array Operations (for both scalar and vector operands)
e.g. Add, Subtract, Invert, Transpose
5. Data String Processing
e.g. Locate, Insert, Delete, Extract, Order, Align.

The User Language should permit the user to allow data items to be used in data manipulation operations merely by specifying their names. In particular the ULS should:

.automatically handle the scaling of items
.automatically handle the conversion of data representations
.preserve significance
.automatically handle conversions from one unit of measure to another
.automatically perform decimal alignment and rounding
.automatically perform truncation and padding of alphabetic data.

Only a limited number of data manipulation operators are provided by current User Languages. Except for decimal alignment, no automatic conversion facilities are available.

Update

"Updating" is defined to mean the ability to change the data base. Changes to the data base occur in two ways:

a) making changes to the data content of items in the data base, i.e. modifying or replacing existing item values
b) adding or deleting items or groups of related items.

The user must have the ability to update a data base. Data is the property of the users within an organization and they must be given the proper tools to exercise their responsibility for that data. If some users do not feel comfortable using the User Language to update data, they can always request that an application program be developed to perform this function. But the ability to update data is a necessary function of the User Language and must be provided in any implementation.

The user should be able to accomplish mass changes, or to update an identifiable unit of data. For example,

Mass change: "Decrease everyone's salary by 10%"
Unit change: "Increase Smith's salary by 50%.

A user must be able to determine in advance the effect of an update operation, as a seemingly trivial update operation can trigger a large chain of updates, resulting in actions which the user did not anticipate. He should be able to perform a pseudo-update, asking the ULS to inform him what the result would be if that update were performed. If the result is satisfactory, he should then be able to tell the ULS to actually do the update, without having to restate his request.

The ULS should have the ability to back-out (i.e. "undo") a series of updates, done by a given user since a specific time, to re-establish the original content of the data base. This is required to enable the user to recover from a sequence of events in which he has requested certain actions, and then realizes he has made an error and wishes that he "hadn't done that" (e.g. a series of actions against the wrong record). Both the user and the DBA should be able to request this function.

Some of today's User Languages such as MRI's SYSTEM 2000 and Informatics' MARK IV do provide update facilities; many however do not, such as IBM's IQF and GIS (when interfaced with an IMS data base). An effective User Language must contain an update capability.

Output

The User Language should support various types of output including various graphical forms (line graphs, histograms, pie charts, etc.) as well as the traditional report forms. The user should be able to select the form he wants,

from a number of defined output "styles", and the ULS should automatically format the output into this style. If additional specific information is needed by the ULS in order to perform this function, the system should obtain this by prompting the user. For example if the user specifies a style which is a report with control breaks and totals, the ULS should ask the user for information as to what totals he wants.

The ULS should automatically format the output onto the selected device type and do any required editing (inserting dollar signs, decimals, etc.) and page control. The edit rules for each data field should be obtained from the Data Dictionary, as should captions. (A given data item may have different edit rules and different captions for different users.) The caption may be used in various ways. For example, in conventional reports it might be shown as "column headings"; in an interactive dialogue it may be a "keyword". The user should have the ability to override these pre-defined captions with different ones if he so desires. Output should be allowed on any type of output device including

- .hard copy - listings, microfilm, microfiche, plots, typewriter terminals
- .visual terminals - CRT's, microfilm display
- .audio response units.

The user should have the ability to examine the quality of his output (i.e. to see if he has in fact obtained what he thought he asked for) and the quantity (i.e. how much output did his request generate) before the actual output process begins. By so doing, the user can determine whether his output is reasonable (i.e. whether he has solved his problem) before commencing an expensive output process. If he has selected far more data than he anticipated, he may wish to apply additional selection criteria against this data base before proceeding to the output stage.

The user should be able to save the output of a request, catalog it for future reference, and reproduce that output at a later time on the same or different type of output device. He should also be able to set limits on the amount of output from his request in terms of volume (e.g. perhaps in terms of pages for printed reports), cost, or response time. Existing User Languages only support output in report form. In the case of IQF and MARK IV captions can be defined as part of the data definition; with SYSTEM 2000 they must be specified with each report requested.

ABILITY TO DEFINE NEW FUNCTIONS

The User Language must provide the ability for an installation to extend the repertoire of the User Language by defining new functions. The user would invoke these functions by name, just as he would those provided as an inherent part of the language. This capability should be available at two levels:

1. a completely new function or operand is defined and added to the language, for example a new output style, or a new data manipulation operator
2. a new function is defined which is a specific combination of functions or operands already available within the language. Examples might be a specific retrieval algorithm, or a tax calculation function. This is somewhat like the "macro" capability available within programming languages.

Some existing User Languages provide a form of the second capability, but none provide the ability to define a completely new function.

COMPREHENSIVE CAPABILITIES VERSUS EASE OF USE

User Languages of the future must provide more capabilities than those available today, if they are to be a viable tool for the user in performing his work. While it is recognized that it is a difficult task to develop a User

Language which at the same time provides a comprehensive range of capabilities and is simple to use, it is not impossible. The User Language must meet the needs of all classes of users. In satisfying the needs of the sophisticated user, however, it is essential that the language is not made too complex for the unsophisticated user. One way of achieving this would be to subset the language, or provide the facility for the DBA to subset it.

It is a pitfall of the data processing profession that we don't know where to stop. Thus in our quest for generality, flexibility, and providing a solution to all possible problems, we end up with something that requires a programmer (albeit not an assembly language programmer) to use. This must not happen with the User Language.

In developing existing products, the various vendors have made the trade-offs between these two factors quite differently. As can be expected, the languages which offer the most capabilities (e.g. SYSTEM 2000) are rather difficult to use, while those that are simple to use (e.g. IQF) are quite limited in scope.

CONCLUSION

Many more requirements for a User Language can be stated; the attempt here has been to describe those which will be of the most significance to the user himself. In the final analysis, the success of computers and data base techniques will be determined by these non-computer oriented users, and only by providing them with a tool which enables them to satisfy their own needs for data directly, will we succeed.

DISCUSSION

CODD:

I believe we should recognize that there are different kinds of users and that there have been many attempts in the literature to identify classes of users. I have noted, however, that most of these attempts have omitted one important kind of user (I have not read all the papers that exist, of course) and that user is what I call the casual user; the user whose job is not bound up with his interaction with a data base, the user who will not learn any stylized language and insists on using English. There have been two schools of linguists at work in the past ten years. One is the programming language school that has developed things like COBOL, FORTRAN, APL, Basic, and so forth. The other is represented by the REL language at Cal Tech and the Converse system at SDC where people have attempted to automate English language through very clever parsing technique coupled with certain other techniques. To support the casual user neither of these approaches is right. I want to stress your statement that feedback from the system to the user, in the language that the user understands, is an important element in moving away from both these schools, which have not succeeded in my view. It is important that the system talk to the user in any language he understands. Through discussion, which is controlled in a sense by the system (except for the opening statement which I think should be entirely free), I think it is possible for the system and user together to come to an agreement on what the query is and what the user has in mind.

KURZ:

Don't you think we should cater to some computer-wide measures like defining the storage structure, data relationships, the manipulation language etc. in an efficient and machine independent manner, so we can even attempt to do something like you propose.

NICHOLS:

No, I don't agree. I believe we're going to cause disenchantment in users of computers pretty fast. I think it's happening already. Maybe they're prepared to pay in terms of machine inefficiency to do things that they can't do today at all.

KURZ:

If we back this far away from the hardware and were given something like this in 5 years time, what are we going to do when we want to change the main frame?

NICHOLS:

Why are we changing main frames? I think we're so absorbed with out own problems that those are the terms in which we're thinking.

KURZ:

Either you'll never be able to change main frames, or you'll simulate forever the system that you ended up providing.

NICHOLS:

Much work has been, and is being, addressed to that area. I think very little effort is being addressed to areas that help the end user.

LEON SMITH:

Where do you see the boundary lines between the kinds of applications which will be done by the user himself and those that will still be done by professional systems designers. Right now we've all been using only very trivial operations and everything else is done by systems designers and programmers. What seems unlikely even in the remotest future is that complete operational systems, such as general ledger or production control, would be designed and implemented totally by the user. I think that we still need professional data processing systems analysts and designers to do that. How far do you see the user going in developing his own system?

NICHOLS:

I actually see him moving away from integrated systems as we know them today. We've gone the integrated system route and we now believe that was the wrong approach. In the future we must have the data integrated so people can get at the data they need. The systems themselves, and the way the users make use of that data, would be in terms of much smaller chunks of logic, which therefore will be easier for the user to handle. I agree it's going to take some time to evolve to that state, and in the next five years there's no doubt that the analysts are going to be doing much of the traditional kind of work.

LEON SMITH:

Do you see a remaining part of the professional world: the computer expert, the data base administrator or whoever? Probably the user does not design his own data bases even though he may design his own programs.

NICHOLS:

Yes.

NORDEN:

I think this is probably the best paper on what really needs to be done that I have read in 10 years, both as a professor and as a member of the IBM Corporation. The questions raised and the comments made are extremely typical of the fact that we listen to the difficulties of doing it instead of emphasizing that it needs to be done. We have models that are artificial oversimplifications of the way the world works whereas reality creeps up with error. These are the things we must learn to cope with. I think you've outlined here not just a user language, but a marriage of applications systems and data base systems so that we can deal with the realities of the vagueness of man.

Data Base Management Systems, D. A. Jardine (ed.).

DATA MANAGEMENT SYSTEMS - USER REQUIREMENTS

E. H. SIBLEY
Department of Information Systems Management
University of Maryland
and
Institute for Computer Science and Technology
National Bureau of Standards

1.0 INTRODUCTION

After accepting the invitation to write a paper for this Conference, I sat down to think about the paper and its contents (an unusual attitude, but one that I try to adopt). Up to that time, I thought that I knew a great deal about the subject:

1. I had been working in the area of Information Systems Requirement Specifications for four years, having participated in the design of a preliminary language for system specification, and keeping aware of other works in the field*.
2. I had read every document that I could find about data base management system requirements**.
3. For a period of five years I had participated in CODASYL's work on the features of data base systems (Codasyl (1969, 1971)) and followed work of others in the same area (Mitre (1973), Welke (1972)).
4. I had followed the technical and scientific literature in the field over the past several years***.

But as I thought more and more of the problem, I became less and less sure of my ground, and more depressed at our lack of technology. I came to the conclusion that if I am an expert, then indeed we are all in deep trouble!

But having confessed ignorance, I still had to admit that I probably knew as much about the subject as anyone else. I therefore started, on a solid foundation of my own lack of knowledge, to structuring the solution to an unarticulated problem. Furthermore, I decided that I had the advantage, after all, of being relatively unbiased.

First, I asked myself whether the industry was becoming too introverted, projecting present technology on to the need, and searching for new problems capable of being solved by old technology: are we, after all, ignoring the "real" user, and merely designing our systems to suit a figment of our own imagination?

Maybe I felt depressed because I am addressing a group of people who are also "experts" in the field, but I suddenly found myself asking questions as follows:

1. If we don't know where we're going, and are doing the wrong thing, why is it apparently so successful?
2. If we are in fact in some sort of a mess, how do we get out of it?
3. But before we struggle out of the mess, we had better know which direction to go: what is the right direction?
4. And finally, is future technology going to push us in other directions?

This paper will try to address these questions, and at the same time rethink the problem posed by that mythical beast: the ultimate user.

* Lynch (1969), National Cash Register (1967), Myers (1962), IBM (1971), Teichroew (1972).
** Guide-Share (1970), CODASYL DBTG (1971), CMS (1971).
*** Steel (1964), CODASYL (1962), Codd (1970), Bachman (1969), Senko (1973).

1.1 Current Implementation

If I were to choose a subheading for this section in the vein of the old Victorian novelist, I should probably call it "success is its own worst enemy". The advent of the computer made it possible to automate scientific and engineering computations (number crunching) and the humdrum operations of business (e.g. payroll, personnel reports, etc.) with very little "design" or thought. Thus, the first phase of computer automation of information systems involved the transliteration of manual systems to punched card systems, the punched card systems to early computers, etc. Everyone has his own favorite horror story of the use of modern computers reprogramming (or even emulating) early systems which have no more sophistication than the original manual system.

The success criterion of information systems in the past has been that they cost less than the system that they replaced. Often, too little thought has been given to what the system is trying to accomplish, and whether this could be done better in some new fashion. Consequently our technology has been one of improving implementation of old methods, rather than working on the design of a better system. Of course, there are some shining examples of improved systems, but all too often these are the exception rather than rule. In consequence, the advent of each good idea in systems implementation has merely reduced cost without necessarily improving the situation or system. Furthermore, there are apparently three mottos in industry.

.Any order is better than absolute chaos.
.It is cheaper to automate than innovate.
.If you change the current system and it goes wrong you'll be blamed, whereas if you improve it you probably won't be praised.

Probably some of the old optimism of the original management information systems buffs led to retrenchment in much of industry. Maybe some of the pessimists spoke too loudly (e.g., Dearden (1972)). But gradually information systems using data base technology are being implemented in industry.

Unfortunately, the concept of the integrated systems still appears to be eluding many of the implementors. I know of two instances of large scale users (one in North America, the other in Europe), who are using a major data base management system with more than fifty totally unrelated or unlinked "data bases". What will we be able to do for these people when they have to integrate their system and convert to a new machine and/or data base management system?

1.2 Building Without Blue Prints

Although the generalized data base management system (as it is being developed in the past few years) has some technological improvements over the operating systems of the past, it provides only one new concept: that of data administration. Most of the goals of generalized data base management are, in fact, associated with improving the lot of the program writer (and consequently that of the ultimate user) by controlling the access and flow of information. Unfortunately, however, although the interface of data definition language and enforced through a somewhat fuzzy interface of data definition language and device media control specification, very few effective tools are being provided for this function.

We have been building new systems based on old ideas, we are being provided with new techniques to store and retrieve data without the decision-making tools needed to provide us with a means of making the right choice. Thus much of the future requirements must be centered on these "better tools".

Finally, being pessimistic about the future, and sure that standardization efforts will not improve our lot in the immediate future, we must consider our conversion or translation of data and programs to future systems; this requirement involves problems of language, as seen by the user, and of continuity of data bases and data structuring techniques. These needs therefore also form a substantial portion of this discussion of generalized data base management systems requirements.

2.0 THE REAL USER OF DATA BASES

In trying to characterize the needs of real users, I turned to a recent excellent article by Terrance Hanold (1972). The main thrust of his article is on the effective design of an MIS, and the fact that this is a feasible present day concept. In this alone, the article is well worth the effort of reading. However at the end, Mr. Hanold makes the following comment:

> "The key to MIS is an integrated data base, not a universal genius-expert and omniscient in everything. As we have noted, each of the functional systems utilizes data and output from other systems as well as data inputs from sources peculiar to its function. From these materials it generates information suited to the performance of its particular function. And this information also feeds back into the data base where it is available as data for all of the other business systems."

He continues to discuss the availability of management information:

> "Some further data will be needed, of course, because executive management considers a different opportunity horizon and a different time span than does operating management. But with this modest qualification the material is at hand to do the job."

Thus Mr. Hanold, a high executive who 'tells it as it is', believes that the MIS is built on tóp of the functional or operational systems, and is integrated with them. He sees the future as being better use of currently available information through better interfaces. Moreover, his specific needs involve:

1. A unified communication system.
2. Rapidly expanding data file facilities.
3. Data coding and addressing to make data randomly and instantly available to every system and user.
4. Adequate methods for structuring data.

The above needs, while slightly paraphrased, represent a high management view of the use of data base systems.

In searching my memory for other papers dealing with the problems of real users, I came upon a paper which uses MIS as the abbreviation for Military Information Systems. This is a paper by Raichelson (1972) discussing the user interface for extremely large systems needed by the military. He breaks these into problems associated with the data manipulation, and how these affect the necessary language interface.

Because of the large volumes of data being used, there is disparity between the application programs which maintain this extremely large base and the higher echelon processing requirements, which require a higher abstraction level, but nonetheless reasonably rapid retrieval. He states:

> "While data attributes and relations can be specified by the user, the software needed to process these descriptions during a data manipulation operation is often interpretive in nature and therefore slow. Since updating a file is a major function of these military information systems, the data files are structured for ease of maintenance, i.e., sequentially, whereas random retrieval facilities, while needed to support an interactive terminal environment, are expensive in terms of the computer resources needed to maintain them."

He then proceeds to show that the advantages of using a generalized data base management system are elimination of unnecessary detail in the application programs, consequently a reduction in the lead time in programming. Furthermore, there is generalization of the security, lockout, job control, and other file processing techniques which lead to the ability for data base administration.

However, he sees disadvantages: such large systems tend to be slow, expensive in the amount of processing during maintenance, inadequate in their provision for data definition and data manipulation languages, need for better data base administration tools, stored data descriptions, and different levels of language to access the data base.

What emerges from these and other discussions of large scale information systems can be summarized as follows:

1. There is a need for the storage of a large quantity of data for operational systems of the enterprise.
2. This data should be structured to reflect the enterprise's method of doing business, but it is liable to change as the technology or decision-making process varies.
3. The output from these operational systems is either information for the operational personnel, or potential agglomerated data for management systems.
4. The operational systems are generally well understood and highly structured. Consequently it is usually possible to treat these in a data processing mode. Because this potentially involves the passage of large volumes of data, the efficiency still matters.
5. Managerial decisions are based on ad hoc rather than prestructured questions. Consequently there is a need for good language interfaces for such high-level users.
6. The cost of processing is dropping rapidly, while the cost of people is still rising. Consequently a presently inefficient procedure may be efficient within ten years, and vice versa.

None of these requirements necessarily suggests differences in our design concepts. In fact, they seem little different from the set of requirements given in the past. They do, however, suggest the need for interfaces for at least two and possibly more types of users. Furthermore, because we are dealing with large, and consequently expensive systems, the transition from one generation of hardware and operational system to another must be reasonably inexpensive. This imposes a further constraint: that the user not find himself locked into an antiquated or constrained system.

There are still two types of user who have not been totally considered in the above discussion. The first of these is generally referred to as the data administrator. His job is to write the data definitions, define access control procedures, restructure the data base as needed, monitor the user operations, and any other administrative or managerial functions that could normally be associated with the processing and control of data. It is not necessary at this point to discuss the difference between terms like "data and systems administrator", nor what functions are normally carried out under these offices. It is, however, important to provide effective tools for these functions, and to realize that the administration interface is one of the most important to the data base management system.

Amongst other things, the data base administration function may need to monitor and augment data base procedures (that is, any procedures invoked by changes due to update or other operations upon the data base). In general, the process of data definition, modification, program and transaction control, test mode, and similar important functions have been too little considered by major system implementers. This is unfortunate, because the data administration function is one in which a small error can produce the most disastrous effects. Consequently control during the data administration function (for example, at program testing) is exceedingly important.

The final user of the data base management system may, in fact, not be considered a user at all, but rather its abuser. He is the systems programmer, who must maintain either the data base management system or its environment. He is therefore in a position to completely destroy both the system and its data.

It seems a recursive problem to protect the system against anyone modifying it. however there should be locks built into the modification method whereby the system maintainer is made well-aware of the effects of his chanqes. Obviously a modularity concept in the initial design of the system is important for ease and control of maintenance.

3.0 REQUIREMENTS IMPOSED BY DATA RELATIONSHIPS

It is unfortunate that we do not yet have a science of datology: the knowledge of data, and the theory and science of its use. At the present time, a few attempts have been made to discuss the syntax of data structures (e.g., the data definition languages, possibly equipped with data manipulation verbs, as exemplified in most data base management systems, as well as Steel (1964), CODASYL (1962), Codd (1970), and Bachman (1969), however almost nothing appears in the literature dealing with the semantics of data. Attempts have been made at understanding relationships between data*, but these are all of a highly general or conceptual nature, rather than presenting a structural methodology.

Not the least of the problems associated with defining data structures is this concept of a semantics of data. As McGee (1972) has pointed out:

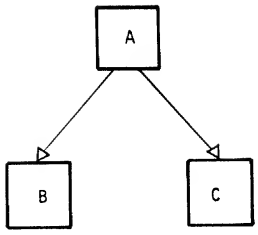
> "The first, historically oldest, reason for describing data is that of aids in the human interpretation of data and in human communication about data. When we look at a number in a printed report, we are helpless in interpreting that number unless we are told what item it is a value of."

This concept of the attribute name and value as a pair has led to the use of set theory for several mathematical definitions of data and its manipulation. The concept of a relation as a set of ordered pairs or of ordered n-tuples has given rise to at least one theory and two working data base systems**, while also producing many artifical intelligence types of implementations (Ash (1968), Feldman (1969)).

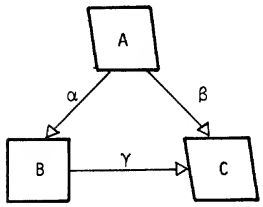
Unfortunately, however, until we either explicitly build data into a structure, or else associate a meaning to a relationship explicitly by using it in a program, we have applied no semantics. This is probably best illustrated by an example. Figure 1 shows three different ways of representing data. The first of these is common to many of the presently implemented data base systems. It uses the concept of a hierarchy between groups of elements. The picture as drawn is rich in semantics, because the implication is that B is totally subservient to A, and repeats as many times as is necessary for a given A. It is equally understood that the relationship between B and C is non-existent except in their common parenthood of the group marked A. It should be noted that all of these semantics are implicit in a way that the data base user is expected to work with the data. Thus the user is expected to know not only the constituent elements of groups A, B, and C, but also the structure and its meaning. This means that the retrieval mechanism in either a host language (in the sense of NEXT verbs) or self-contained (in the sense of COUNT verbs) is well understood. As an example of this confusion, any NEXT implies an order of threading through the instances of the hierarchy, whereas COUNT implies that there is a knowledge of scope of the verb (i.e., we are counting the groups of B associated with a given A, and not all B's within the data base).

The graph type of structures are more explicit in their naming of the relationships between the groups of elements, but generally they have less semantics associated with the explicit relationship. This is one of the reasons that graph structures have been difficult to retrofit with self-contained features. In fact, most of the current implementations of a query or other self-contained facilities for graph structures, such as the CODASYL DBTG (1971) report implementations, have defaulted to structures which are essentially hierarchic in nature. This means that the present implementations of a graph structured query language are really no more powerful than those of hierarchic systems.

*Codd (1970), Bachman (1969), Senko (1973), Mealy (1967), and Ash (1968).
**Steel (1964), Codd (1970), Childs (1968).



(i) Data Hierarchy



(ii) Data Graph

A $\{<a_1a_2>\ <a_3a_4>\}$

B $\{<b_1b_2>\}$

C $\{<c_1c_2>\ <c_3c_4>\}$

α $\{<a_1b_1>\}$

β $\{<a_3c_1>\ <a_3c_3>\}$

(iii) Data Relationships

FIGURE 1 THE SEMANTICS OF DATA

The data relationships have essentially the same power as graph structures, except that their physical representation is generally much more clumsy. Some of the dependencies have led to concepts such as Codd's "normal forms", which factor out the name of the element and allow special operations such as natural-join on simple relations. The problem with these systems is that they have absolutely no semantics associated with the relationships (other than those of formal mathematics). Consequently it is easy to join together two relationships which have little or no semantic connection, but have commonly named elements. As an example, it would be possible to join together two relationships which have the common element termed HEIGHT even though one of them was in a relationship termed MOUNTAIN, and the other in the relationship termed PEOPLE. This is not to suggest that such an operation is necessarily incorrect, it merely suggests that the semantics associated with a more explicit structure, such as that of the hierarchy, might suggest that the two different groups relating MOUNTAINS and PEOPLE were not explicitly structured to make this a sensible question.

Of course, the use of semantics would help decide whether unexpected relationships should be allowed (and presumably the relational structures allows us unexpectedly to ask questions like: "Give me the common pair of names of PEOPLE and MOUNTAINS which have the same HEIGHT"; if we so feel like asking).

Thus we see that one future need of our systems is a more careful statement of the semantics of our data, if not this, at least a careful study of the interaction between the implicit semantics and the data manipulation verbs.

3.1 The Need for Data Restructuring Methodology

The problem of whether a data base is capable of restructuring or not is presumably philosophical. I therefore do not hesitate to discuss the problem of modeling data as one view of information structuring.

In dealing with the use of data, we are presumably dealing with some "real world" concepts. As scientists, we believe that such a world is governed by (relatively) immutable laws, some of which we already know and can formulate or codify, and some of which are still unknown. In order to understand and deal with this world, we therefore model it, and hope that the model will be sufficient for our needs: naturally, it is not, and must constantly be modified½ Thus we must look at our model of the world as a present interpretation. This could be represented as shown in Figure 2.

The figure is intended to convey the concept of the real world which is being abstracted in a special way to suit our present needs. Thus a person dealing with a particular branch of physics such as quantum mechanics is unlikely to need information on medical jurisprudence to further his research: the scientist is interested in some current model of the real world as he needs it. Thus there are two levels of abstraction: the summation of all knowledge about the real world, expressed as some "best" model, followed by the model that particularly suits a worker's need (and this may be a sub-model, or some deliberate "bending" of the model in order to discover some new relationships). The scientist's task is now to manipulate his current model, or abstraction, to solve the problem, or to discover new information about the real world (i.e. universal laws).

A parallel to this in our normal data structuring for scientific and business applications of data base management systems involves a conceptualization or modeling of the real world as a grouping of data names and relationships which is, at that time, considered the best model, or the "way of doing business". For the present time, this will be termed a conceptual or super-schema. No formal definition will be given for this, but it is hoped that the reader will understand it as a current 'best model' described in some data definition language.

The user of a data base management system is expected to deal with some subset of the total model of an enterprise or scientific endeavor. He therefore abstracts that portion of the super-schema to define that subset that he sees necessary for his particular purpose. This subset will be termed a virtual or sub-schema. It is within this sub-schema that the user language or programs interface. Naturally, it is assumed that there is a well defined mapping between the super-schema and the sub-schema. This is illustrated in Figure 3, which also introduces some mapping rules that will be used later. It is assumed that, in

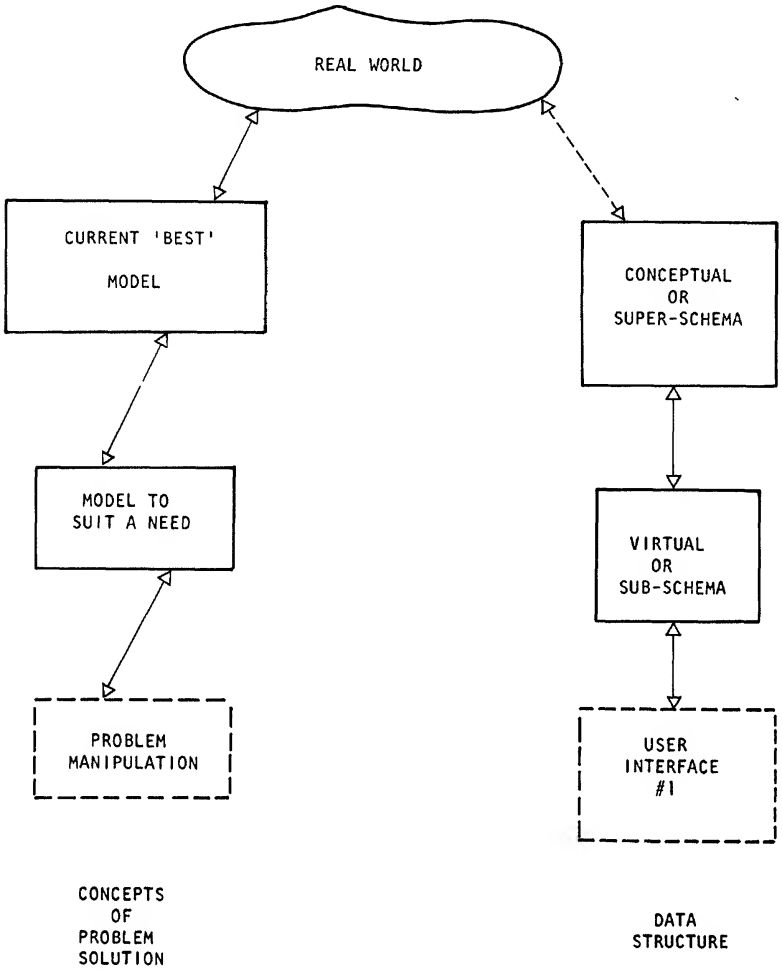
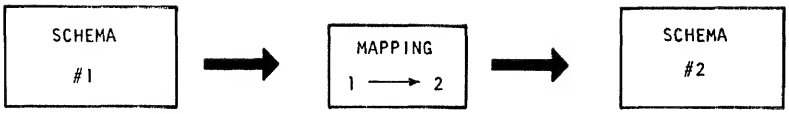


FIGURE 2 DATA STRUCTURING AS A MODEL OF THE REAL WORLD



1. SCHEMA #1 DEFINES SCHEMA #2
   BY MAPPING 1 → 2.
2. IN GENERAL THERE IS NO INVERSE MAPPING.
3. IF THERE ARE OTHER MAPPINGS
   BETWEEN SCHEMA #1 AND
   SCHEMA #2, THEY CANNOT
   ALL BE INDEPENDENT.
4. THUS MULTIPLE MAPPINGS CONTAIN
   DERIVABLE ELEMENTS.

FIGURE 3 MAPPINGS BETWEEN SCHEMAS

general, something is lost at the time of mapping. Certainly, no additional information can be generated by the mapping, and therefore Schema #2 will normally represent a potential loss of information compared with Schema #1.

The model, as derived so far, has not discussed the way that data is currently obtained or available. Unfortunately, it often happens that the data is not immediately available in the particular form that the user needs it (as viewed by his sub-schema), nor is the information necessarily stored in the conceptual fashion according to the super-schema. In fact, the real world data base probably represents another, different abstraction of the super-schema, which we will call a logical-schema. The relationship between these three views of the data structure is shown in Figure 4. It is important to realize that at the present time most data base management systems provide a logical schema interface by providing data definitional languages. They may or may not provide an external virtual schema by either:

1. Using an explicit definition (e.g. the sub-schema of DBTG, or the SENSEG of IMS).
2. Implicitly defining only some subset of the total data base in the data division of a COBOL program, or only accessing certain portions in a procedural manner.
3. Identifying types of users, and consequently allowing a user to access the logical schema through a "screen" which represents his virtual schema. This is the method which must generally be used for systems supporting self-contained or query features.

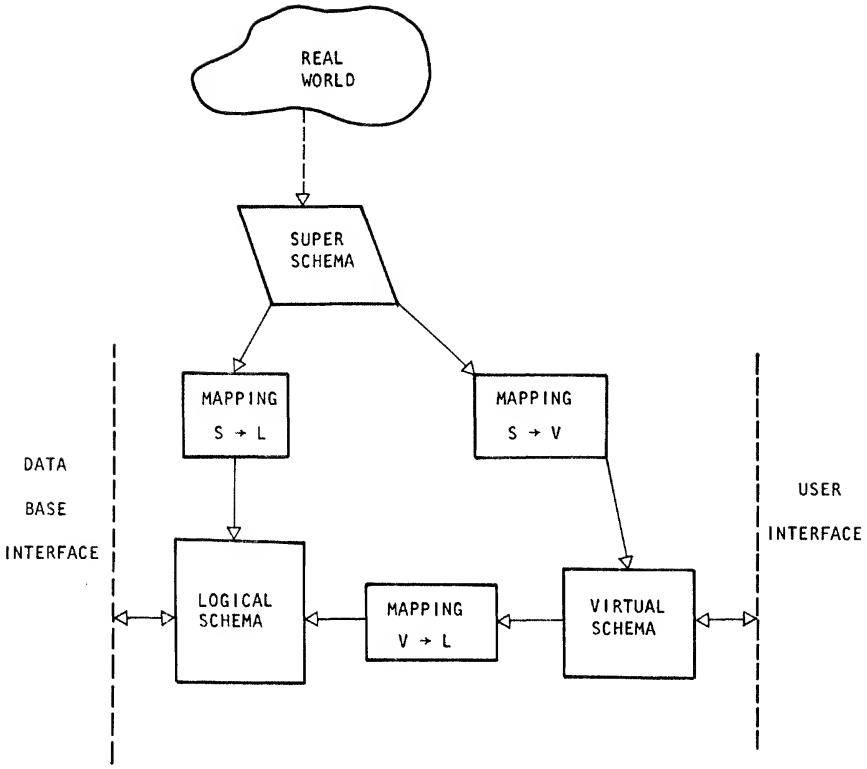
The relationships between the mappings now becomes important, because it will be necessary to determine some of the compatibilities or incompatibilities between a user's needs and the data and relationships physically stored. Naturally, a user cannot expect to receive information that has not been physically implemented, but it might be reasonable to allow him to ask for it, and be told that there is none, or to write programs on the assumption that at some future time it will be available as one of the elements. It is therefore important to see that in going from the super-schema to the logical schema there are two paths. These may be represented as follows:

> If MSL represents the mapping from S to L then the path from super-schema to logical schema is either MSL or the concatenation of MSV and MVL. Now by the rules suggested on Figure 3 these two paths cannot be independent. This means that it should be possible to generate MVL given the other two. This would then make it possible to interface user programs with transformation programs generated from the data structures, and consequently allow an entirely new and important dimension of program and data independence.

Another important reason for investigating the relationships between a super-schema, logical schema, and virtual schema is the fact that any one of them is likely to change. Obviously the virtual schema, as the users' interface, will be promulgated by many different users having different requirements. In the same sense, the logical schema, though changing less radically may nonetheless vary with time due to the data administrator's comprehension of better ways of doing his particular job (one of which involves the structuring and restructuring of the data base). Finally, because the corporation may be changing its way of doing business, (or better models of science are being developed), the super-schema must (irregularly) be undergoing change.

Presumably every time that there is a change, the new schema is not totally different from the old schema. It may contain additions, new explicit relationships, or possible transformations of the old relationships. If the transformation from an old schema to a new schema can be stated explicitly in some formal manner, then presumably they represent another type of mapping (a time mapping). If we had a description of techniques for mappings, then time maps would allow us to access old data bases with new programs and vice versa. In consequence, we would have an ability, implicit within our systems, of ease of conversion of data structures.

Naturally, this does not immediately allow us to transform to new storage structures, but this represents the problem next discussed.



NOTE: ARROW DIRECTIONS ARE IMPORTANT BECAUSE OF NON REVERSIBLE MAPPING.

IF MAPPING S → L IS REPRESENTED AS "MSL" THEN:

MSL SHOULD BE DERIVABLE FROM MSV AND MVL

FIGURE 4 THE SUPER, SUB AND LOGICAL SCHEMAS

3.2 Storage Transformations

The current relationships between data structures and storage structures represent a problem whenever we are faced with system conversion. As a result, some research effort is now underway in the area of defining data base storage transformations, with a view (ultimately) to providing better mapping methods, more efficient structuring, ease of system conversion, better possibilities for networking between similar and dissimilar systems, etc. Much of the work discussed at recent SIGFIDET workshops has dealt with this and similar problems.

A task group of the CODASYL Systems Committee, working on stored data definition and translation, reported (CODASYL SDDTTG (1972)) on their work and similar efforts in the field. Furthermore Sibley and Taylor (1973) give the following conclusions:

> "1) The formalization of the description of the storage structure of modern information processing and generalized data base management systems can be used for better communication of methodology. Presently this description is verbalized in users manuals. The description is usually fractionated throughout manuals and sometime is incomplete.
> "2) A formal stored description could be used by the system itself for many processes which are now difficult or impossible. Some examples are: the process of reorganizing the data base (i.e., the collection of useless or redundant data), the process of restructuring the data base, and the ability of a system to generate its own access mechanism based on the knowledge of the logical access paths and their physical relationships.
> "3) A formal description of data could be used by other systems for inter-computer communication. The use of computer networks, possibly with distributed data bases means that one computer system may need to utilize data stored by another. The data may be made available in one of several ways from total reading, with translation, of the foreign base, to generation of query language requests in the foreign language."

Further discussion of the problems of multiple data description, physical data description, and mappings is given in McGee (1972). McGee in this and other works has discussed the use of a "correspondence definition" between a user's program and the data base. In many ways this resembles the admixture of the mapping between virtual and logical schema in conjunction with some portions of the physical mapping. It would allow one program and language to access a foreign file (e.g., a COBOL program to read a file generated by FORTRAN).

Thus the need for multiple user access to data base, of different language interfaces, and of different users' views of data structuring, as well as the needs for distant access through networks, all require a technology involving the definitions of explicit mappings between data structures, and also mappings from a logical data structure to a storage structure, and onto a physical storage device. Such a technology would allow us to restructure and reorganize data bases, transfer them from one physical system to another, and allow multiple types of language interfaces. Obviously, however, there is no system which currently allows this, nor is much of the technology yet developed.

4.0 THE USER LANGUAGE INTERFACE

The problems of the user language have been discussed in Section 2, while looking at users' needs. In accordance with Hanold (1972) and Raichelson (1972), we may conclude that there is indeed a need for multiple language interfaces. These interfaces must include at least the following:

1. A transaction-program-oriented or parametric-user interface, which allows particular predefined programs to be run, possible with control parameters and special inputs. Naturally, this interface should be as simple as possible because it deals with a class of user who has no real interest in learning about computing, but merely "has a job to do". The education of these people,

who may be changing their mode of work, could prove expensive, especially if they are prone to error during this education period. The advantages of an interactive medium, potentially using as close as possible to natural English language as an interface, is therefore advantageous. The "human engineering" of this interface is all too often ignored by systems builders.

2. A higher level language interface, which requires some education but can be taught to engineers and scientists without excessive education and knowledge about the trials-and-tribulations of computer programming. Naturally, this language requires the user to know something of the data elements, their structures and relationships. However it should definitely be as "non-procedural" as possible.
3. Several programming language interfaces, allowing several types of users (e.g., engineers, business, social scientists) to interact efficiently with large volumes of data. This would typically, at present, suggest a COBOL and FORTRAN interface, but presumably other user varients of compiler languages may be needed in the future.

The whole problem of program and data independence now resurfaces in the guise of a discussion of the data manipulation languages. It is obviously possible to obtain more or less independence in several different fashions. The first is to expunge a "procedurality" or "knowledge of the structure" from the program itself. This means that such verbs as "NEXT", and possibly "COUNT" must be removed. An alternative is to make sure that these types of verbs are only used in a qualified sense such as "NEXT, ALPHABETICALLY WITHIN GROUP OF THE SAME TYPE" or "COUNT WITHIN THE SAME HIERARCHICAL PARENT" or "COUNT, WHEN NAME EQUALS JONES".

Unfortunately programmers, and even real people, like to make terse statements, where their knowledge of the system (or their assumptions about the understanding of other people) might otherwise seem ambiguous. To insure well defined questions assumes the correct understanding on the part of the requester, possibly with a dialog, which may seem annoying in general conversations. As an example, consider the following conversation with an erstwhile friend:

"What do you think about the Watergate incident?
"Well it depends whether you mean the Watergate Incident recently."
"Yes that's what I mean."
"How do you mean what do I think about it?"
"Do you think it will affect the next election?"
"It depends what you mean by affect."
etc.

Such a conversation only makes us conclude that we are talking to a bumbling idiot, but presumably it insures that we do not have the following sort of a question and answer:

"What is the relationship between wood and Louisa May Alcott?"
"Shaker Furniture."

I suppose we've all been amazed at the ability of experts on certain quiz programs to come up with this sort of convoluted answer. The "arguments" for the answer are that Louisa May Alcott used to go to Harvard, Mass, accompanied by her transcendentalist friends. The transcendentalists were a relatively communistic philosophers, of the same variety of communal concepts as the Shakers, who produce beautiful wooden furniture. Now perhaps this proves that any complicated question, or question which is answered by a complicated process requires better than a simple response: one which shows how it is obtained. The trouble, of course, is that nobody likes a smart aleck (system). A system which keeps on insisting on clarification (possible even at the twenty fifth time of asking the same question) is effectively of no use. However the other side of the coin is that a system which gives you an answer which may be right or wrong is probably equally unuseful. At present, we have no good way of checking ad hoc requests: Is it because we are often ambiguous in our definitions?

If we really intend to allow users to ask complex questions in an ad hoc fashion, presumably we need some debugging interface for anything but the simplest questions. The only alternative seems to be to limit ad hoc users to a highly simplified view of their data and possible therefore to very simple requests.

Possibly, also, we must protect the user against his own stupidity: for example, if he asks, in the national data base, for a list of all people who are brown haired and between 15 and 70 years of age. Maybe a question should always be answered in two parts, the first providing a cost and run time estimate, the second the answer. If the system determines, say after attempting to answer the query for one minute, that is 90% completed, it should continue, but if it is only 'well-started' it should interrupt, print an estimate, and ask whether to continue. Such introspective machines may, of course, be annoying, but may be necessary.

5.0 DATA ADMINISTRATION TOOLS

There are probably four principal generalized functions of data base management systems:

1. To maintain and manage the retention of an expensive resource.
2. To make it available to users and user transactions on authorized request.
3. To reduce the cost of a disaster involving loss of the data.
4. To facilitate changes to and modification of hardware and programs.

If we define the role of a data administrator as being the care of all machine processable data within the organization, then each one of the requirements intersect with this officer. Consequently the tools to provide him with means of carrying out his function involve methods of monitoring and controlling the functions of data base management. Each of these will not be discussed separately.

5.1 Managing the Resources

In order to manage a resource, it is necessary to know something of its use. This is probably one of our greatest problems in the information systems area: a lack of knowledge of how, when, and where data is used. It affects the design and operation of all systems. What, then, are the tools necessary to aid a data administrator in pursuing this management function? There appear to be two partially developed methodologies as follows:

1. Providing a data element dictionary. The difficulty here is knowing how to develop the dictionary, and what to put into it, followed by how to use it to structure the data base. If we are starting with a new system, then the task is one of gathering users' needs in some consistent fashion. The discussion of techniques for doing this may be found in references 1 through 5. In general, these entail determining the names of elements as they appear in the input and output of information, complete with timing and frequency information. Thus a minimal definition might be verbalized as follows:

   "A personnel report is produced weekly. It consists of names, grades, salary, and action for all transactions that have occurred within the personnel office involving modification to the personnel files. It is generated after the last transaction on Friday. Approximately two hundred and fifty entries will be made each week."

   Presumably from this information, the data administrator can start a directory or dictionary involving the names of elements, how often they are used, and how they are grouped when used. Part of the needs of a data administrator obviously devolve upon methodology for determining good structures, given such information, possible with simulation programs which allow him to test out the efficiency of various structuring techniques both at the logical and physical level.
2. Using data gathering (statistics) utilities to determine data element utilization. Once the data base has been structured, and is running, it is

* Lynch (1969), National Cash Register (1967), Myers (1962), IBM (1971) Teichroew (1972).

possible to gather statistics on the use of data. Sophisticated routines should be able to determine the number of accesses of the data base, redundancies of retrieval, and degree of garbage within the files, which allows the data administrator the information for restructuring or reorganizing the data base. Reorganization presumably can be used without any changes to the data definitional programs to provide better data accessing either in the small (by reducing garbage) or in the large (by utilizing new storage structuring techniques). Naturally the latter requires more sophisticated methodology than presently exists, although work in this area has already been undertaken (Senko (1973), Severance (1972)). Amongst other things, this ensures that unnecessary redundance of data does not occur.

There is one area of data redundancy which requires special control, and that is when it is realized that redundancy may allow more efficient running of the procedures, but might involve a need for multiple update given a single transaction. This is one of the best reasons for the inclusion of a "data base procedure" concept within the design of data base management systems. The assumption is that the invocation of the procedure is automatic, being triggering by some action, such as an update. In the case of duplicate data, both cases of the element data definition would require concurrent update; i.e., on either file being updated with a new value of this particular item, invocation of the other file update process would be automatically accomplished. This does, however, place extra overhead on the system (and consequently requires special investigation of the tradeoffs of redundant data).

The problems of effective file creation include decisions on data conversion, compression, and validation of input. Tools for file creation would therefore involve methods for estimating size, sampling of files for compression and efficient storage techniques, and ease of providing data conversion and validation routines.

With the advent of better output devices, the ability to provide more efficient multi media outputs (such as video tubes showing graphs, potentially with audio response commentary) becomes more and more feasible. Presumably multi media outputs involve use of current or future operating system interfaces, but they will require planning on the part of the user in ways of effectively utilizing them. Once again, the human engineering aspects must be considered.

The other resources associated with the data administrator are those of the data base procedures, both those designed for his use alone, and also for the general user libraries, etc. Good system interfaces are absolutely essential for ensuring ease of operation. Some systems have made it extremely difficult to update programs (and test them, as discussed later) within the data base environment. The ease of redefinition of transactions is therefore an important area in future data base management system technology.

5.2 Control of Access to the Data Base

It is very difficult to tell where access control ends, and where integrity begins. Consequently this section and the next will overlap. Obviously any breach of security in the sense of accessing or updating a data base potentially causes its integrity to be downgraded. The first problem is therefore that of the security of the entire system.

As they presently exist, data base management systems operate within the environment of an operating system. Operating systems as they have been designed, and currently are implemented, are badly lacking in effective control of access. Consequently, any system built on top of an operating system is itself insecure. With the expenditure of large efforts in the near future, it is anticipated that the security of operating systems will gradually improve to the point where they no longer represent the weakest link. At this time, better data base methodology will be required.

The problems associated with control of access in future data base management systems are as follows:

1. The data base administrator may wish to have control over the relative scheduling of programs operating within the data base environment. Either he must have this control methodology available to him (presumably with all of

the protection that previously existed within the operating system), or else he must rely on the operating system delegating him the responsibility for the assigning of program accessing priorities.

2. The file management techniques of the operating system must be utilized by the data base management system, unless the security is to be totally in the hands of the data base management system (and consequently of the data base administrator). Presumably all users will not have the same right to access, and consequently the control of access must once again be vested in the operating system. However, there may be more difficult access rights than merely that of a right of a person to a particular piece of information. As an example, in national statistics it is normal to apply some random error to sensitive data. This is to ensure that the user may not take an average (normally allowed) associated with a single piece of data, or between two numbers, where he knows one because it is his own. (e.g., if I know the average of your and my salary, and I know what I am earning, I know therefore exactly what you are earning). In the past, these types of problems have either been considered too difficult to solve, or outside the scope of the data base management system. Such must not be the case in the future.
3. Other problems of control and access are associated with the checking-out of programs against live data. This is discussed further in the next section.

Unfortunately, apart from the provision of validation techniques, very little has been provided or discussed for maintaining effective quality control over data. Some ideas have been discussed (e.g. Harrell (1972)), but few have been effectively implemented in any but the most sophisticated systems. Statistical techniques, involving internal validation over confidence limits, validation at each access, and more sophisticated relational testing are a minimum for future systems. At the best, we should be providing some sort of inventory techniques: as an example, we could take stock of the data base, in the normal meaning of 'inventory', on a regular basis, as well as by allowing a system to request regular verification of some portion of its data by interacting with an input station; e.g., a personnel department may be requested to verify that user number 15723 has the name SI7BLEY (either because the seven was detected as being an unlikely character, or because it happens to have become "my turn" to be checked).

5.3 Minimizing the Risk of Disaster

Data integrity is threatened whenever the machine (or one of the users) goes wrong. This may be temporary, in the sense that the disc controller cause an error, or a person makes a mistake, or permanent in the sense that the data base blows up, or the user sets about destroying important data. It may also be unexpected, in the sense that an apparently debugged program suddenly shows its real colors, or an unanticipated sequence of operations causes lock-out, or a new concurrence situation arises.

There are really three parts to the problem: the first is to provide traps, which minimize the chances of an incorrect set of operations; the second involves the detection of a potential error condition, and either warns the operating staff of potential threat or endeavors to rectify it; while the third provides procedures so that any damage is not permanent.

Most systems provide methods for recovery from a detected error. These involve taking snapshots of the core (normally called check points) so that programs may be restarted after hardware failure, providing either full scale or partial copies of the data base so that the entire set of information may be brought back in the event of a major storage failure (this normally also of course involves logging transactions, and possibly keeping information about updates, before-and after-images, to the data base), and finally procedures for rolling back the system so that an incorrect procedure can be expunged.

Not the least part of the problem is the fact that any error which is undetected for a period of time may cause a general "pollution" of the data base due to secondary changes. This occurs, for example, when an incorrectly updated item is used to generate other outputs or changes to the data base. As an example, the result of the change to the price of a single bolt could affect the price, apparent profitability, and output quotations for a very large class of

objects using such an component part.

Not only must good techniques be developed to aid the data administrator in recovering from these errors, but obviously aid is needed in determining what actions should be taken, and how often they should be taken. Very little, in fact, has been done to aid the data administrator except for results given in a recent thesis (Sayani (1973)). The data administrator needs answers to questions like: Do I take partial data base dumps? How often should a complete data base dump be taken? (It is assumed that the cost of dumpting of a large data base is not negligible, and consequently should not be done at the whim of the administrator). Should programs be allowed to provide their own roll-back procedures? How often should check points be taken? Would it be cheaper for me to run with a duplicate data base at some remote location?

The provision of methods for minimizing error by early detection are often hardware oriented. Such methods involve hardware and software devices such as parity checks, vibration and noise sensors, and redundant circuitry. There are, however, potential threats which can be minimized: these involve protection against duplicate updating of the data base, control over program modification, provision of program test techniques, and detection of potential lock-out (or deadly embrace) situations.

The detection of concurrent updates will normally be determined by the scheduling modules. In this, there is a great deal of similarity between concurrency protection and lock-out prevention. However the provisions for sophisticated data base procedures makes it difficult (if not impossible) to determine what resources are likely to be needed by any one program or set of programs at any one time. This is because the program can cause invocation of a data base procedure in a previously unanticipated fashion, which consequently produces unexpected scheduling of programs, and potentially a lock-out situation. This is one of the conditions which can be solved by allowing the condition to occur, but ensuring that it is detected; then it can be corrected by automatically backing out one of the resources, and holding it for rescheduling after completion of the other (contentious) procedure.

Other methods of reducing errors due to concurrency involve holding some, or all, of the data base until a particular program or procedure ends. Although this may be a reasonable method for small or relatively short programs, it could lead to excessively long locks on large portions of the data base in real life situations (with long jobs in the mix).

Another important concept is that of providing testing techniques; test programs (and potentially some production programs) do not make permanent updates to the data base, but either make temporary changes, or record the fact that a temporary change occurs in an extra temporary file. This procedure allows check-out of potentially dangerous programs (i.e., those which could violate the integrity of the data base), and also allow certain programs to operate in a "temporary change mode." This latter mode is one which can be effectively used in industry when several users are vying for a particular set of resources which are currently available to them. As an example, several different schedulers may be selecting from a common pool of machine tools to provide an optimal job shop environment. Modification to the schedule must presumably not be made until a sign-off is given by affected parties. However, other changes (such as future schedules) should take cognizant of the anticipated change, while being warned of the fact that this may only be temporary.

5.4 Minimizing the Effect of Change

At times it seems as though the data base management system has the supreme irritation of being a pimple on a pimple. It exists within a operating system. The operating system exists on hardware. Unfortunately, the real user is therefore in an extremely uncomfortable situation. First, he intersects (possibly through a programming language not of his choice) with a data base management system. The data base management system exists within the environment of the operating system, and the operating system sits on the top of some hardware. Thus the real user may find a change due to adoption of a new concept within the given data base management system, a change to a new data base management system, a

change because the operating system no longer supports the data base management system's previous interface, and a change because of a move to new hardware.

Although the ultimate user may be relatively buffered from some of these changes due to systems programming and other functional staff, be nonetheless bears the brunt of the cost.

Standardization of languages has been the first moderately successful attempt at making it easy to go from one vendor to another. But this has only shifted the interchange problem, because the data base management system is generally not transferrable, and even if it is, the data may not be. Thus, there seem to be at least the following incompatibilities:

1. Most manufacturers provide an operating system as the principal interface: a standard operating systems command language would surely aid in reducing irritation at this level.
2. If the data base management systems themselves were implemented in a standard language, then they would be themselves more transferrable. This of course requires the good graces of the hardware manufacturer in providing a reasonable interface. A common operating systems command language interface would obviously aid in resulution of this problem.
3. In order to be able to utilize the same data even though going to new hardware devices, either standard storage structures or data translation techniques are needed. These concepts have already been discussed in the article.
4. If we may assume a geographically distributed data base, it will be necessary to have some standardization of the telecommunication support, or at least their interfaces.
5. On the assumption that most systems are somewhat changeable, or at least evolutionary, then it is necessary to have statistics gathering packages which allow for reconfiguration of input output devices, core, and channels for an efficient overall operation. Naturally, it may be assumed that such packages will ultimately be provided within the operating system environment, but at the present time they will probably be provided through the major user: i.e., the data base administrator.

6.0 FUTURE TRENDS

It would be meaningless to close an article on the trends in data base management systems without considering the effects of new hardware on our technology. Substantial reductions in cost of hardware are still predicted to continue. This means that the greater part of our hardware cost will be dropping by a factor of ten in the next seven or eight years.

Consequently much of the processing costs, which are already fairly small, will become negligible. At the same time, the cost of storing data for one year on high speed memory, which is still of the same order magnitude as the cost of collecting it, will presumably become relatively small. All of a sudden, our cries of "efficiency of storage", "reduction in processing costs", etc. will be secondary to the cost of developing the system, and the wages of the people who are using it. Possibly this is an over optimistic view, but the arguments which held off the application of associative memories (i.e. that they cost ten times as much as conventional hardware) will become invalid (ten times nothing is still nothing). So maybe we can afford the luxury of better relationships in our data, and highly parallel processing for retrieval.

So with due reference to new hardware we may say with Shakespeare:

"The fault, dear Brutus, is not in our stars,
But in ourselves, that we are underlings."

DISCUSSION

LOWENTHAL:

This is a vendor's response or at least a vendor's viewpoint of what I have been hearing. This is in reply to both you and perhaps a bit to Pat Nichols. Data management originally was that branch of information retrieval that dealt with trying to process large amounts of data at great speed, as opposed to very flexible queries with great semantic power. We (especially Pat Nichols) have been talking about combining those two so that you have the advantages of both. You have been saying in tandem that we should not be worried about efficiency and that flexibility would become increasingly important as hardware costs go down. I take exception to that particular view. In our experience the requirements for greater efficiency seem to be increasing with centralization of data, with the increasing size of networks, the number of transactions per hour, and multiple billions of character data bases. The needs are outstripping the advances in hardware. As a company we would love to be able to direct our resources to making our query language more flexible but we find that our wish list is more in the area of providing more primitive kinds of functions to permit greater transactions rates.

SIBLEY:

I guess it is a matter of time horizons to begin with, and I will live with history for the moment. First of all, when I talk about user requirements of this nature I am talking in the five to ten years horizon; I am not talking about right now. Consequently I take history on my side in the sense that the prices of hardware have been dropping approximately ten times every seven to ten years and presumably the same will be true of large-scale memories. We're on the edge, I think, of another drop off on the cost of CPU's. I therefore don't see the need for worrying about fast retrieval becuase I think we will get it anyway, and I seriously don't believe that that is going to be the problem. On the other hand large industry, large systems users, often take eight years to develop their system and we can't wait now to know what the system is going to be like in eight years. We can only make a presumption that it is going to be one where the costs are so low that the efficiency starts to be uninteresting.

STEEL:

People don't seem to know what the word 'efficiency' means. We sub-optimise, we worry about making this little piece of the thing run fast or that corner being cheap and we don't look at what the problem is and what we are really trying to be efficient about. We are trying to be efficient about running our business, not how fast we can pump transactions through a piece of hardware. If you want to examine the cost-benefit balance of these things, it turns out that if you save people time, that's where you really save the money. I think this point ought to be a little better understood by the vendors.

LOWENTHAL:

The kind of experiences we have had indicate that users want to learn how to get around the flexibility of the system in certain cases in order to cut down the number of I/O operations. For instance, in System 2000 we had validation at the item level so that when you moved a record from the user-work area to the data base we look at each item and examined it very carefully. The feedback was: isn't there some way we can get around all that; just move the whole block over at once. We considered this to be a regression in terms of capabilities and yet many users need that (or at least feel they need that) in order to reduce the transaction times to a minimum. I feel that there are a number of applications out there which require that kind of efficiency.

STEEL:

I sympathize with you as a vendor up against unsophisticated users and maybe one of the things you have got to do is make your users much more sophisticated. Generally speaking when you say user you are not talking to a real user. You are talking then to a computernik and that isn't a real person.

LOWENTHAL:

We are talking to the customer. The man who is paying.

STEEL:

Usually the man who is paying has delegated responsibility to the data processing manager who doesn't really understand what users need. He is more worried about the bits and pieces.

LOWENTHAL:

I am willing to agree that the customer is often at odds with the end user because the customer might be providing a service. He might be in charge of maximizing through-put on his limited resources, his small computer. Nonetheless that is a very real need just as the end users needs are real, and cannot be ignored.

KAMERMAN:

From the point of view of requirements we tend to say, yes, indeed, hardware prices are going down, but for all that we still have some problems. For example, electricity travels about ten feet (SIC½) in a nano-second and you get into certain areas of performance where one would like to be able to measure it in terms other than furlongs per fortnight or something like that. As we talk about particular functional requirements we have four competing perpendicular requirements: function, performance, recovery and security. After you deal with things like Pat Nichols' user language, the fully recoverable data simultaneously accessable with complex structures, non-redundancy of data with full security, the flexibility of making all of these changes all of the time, what we have done is eaten up everything in the hardware. The only thing that we don't have any time left for is the execution of an application program. It seems to me that as we state requirements we have to talk about things like configurability or selectivity amongst these things. If we try to put all of these things into a single system, that system isn't going to do productive work except for the computer scientist who likes to look at all the nice things he has put in it. I think we should very carefully state the selectivity requirements that we have to make a system useful, as well as having all of these nice things. Otherwise we are going to eat up the whole world with our data base systems.

SIBLEY:

I personally believe we can go a long way past where we presently are and still not have eaten up all of our resources. We have systems which make primitive attempts at doing all of the things you have mentioned. There are going to be some systems with special requirements. For example National Security data presumably does still need to have some sort of higher security. The stuff that we want to put on, say, a personal data base is somewhere in between. Presumably you don't want this to be openly accessable. Finally at the other end, say, in a library book list, we want everybody to be able to access. There are differences in security, and we might very well have to tailor our needs but on the other hand I see no reason why we shouldn't go a great deal of the way towards our goal. We have in fact systems which are doing a good job of 20% or 30% of where I hope to be in five years time. I don't happen to think that we

are going to be constrained. I think we are going into a era of multiprocessors. If we don't get into an era of multiprocessors, we are going to be into an era of lots and lots of minicomputers and we'll solve the problem by breaking it up.

KAMERMAN:

Does your reply indicate that you don't feel that a selectivity of function is important in a major data base system?

SIBLEY:

It depends whether we are talking about generalized systems or non-generalized systems. If we had something like Pat Nichols stated in her requirements, then we have a real problem on security, or, alternatively, we had better have some good tailorability of what a user is allowed to access If by that you mean selectivity then, obviously, yes, I think it needs selectivity. I flipped between the network concept and the utility concept and I don't know which one is going to win yet. If we have utilities obviously we have to look in terms of a security system which does it all, and maybe which you can short-circuit at times (because for certain requests you don't have to keep on asking whether you are allowed to look at it or not).

METAXIDES:

I would like to comment on this discussion about efficiency. I am rather disturbed that the word 'user' is being used to refer from time to time to one class of the various classes of users that have very carefully been mentioned by various people. I think you yourself (Sibley) described various classes of users. Later on in your talk you are talking about one class of user and that disturbs me. As I see it there are these various classes of users, one of which is the non-programming user. That itself can be broken down into several sub-classes, but the non-programming user is interfacing with data. Now where is this data coming from? Surely that data has to be up-to-date otherwise there is no point in making decisions on the basis of the information gained. That data surely then comes from the day-to-day routine massive volume operations that reflect the business itself. In my view we should not forget that that data has to be collected and I think that there will be for a very long time, room for the programming-type interface for developing routine day-to-day operations and applications and for building and maintaining the data base. This then gives the base (as the name implies) for the non-programming user to get the information that he may need, provided that there is an intelligence-infusing system to deal with the data in the data base. I think we are taking a backwards step in reviving the question of whether programmers will continue to exist or whether data base management systems should be entirely developed for the non-programming type person. I thought that question had been settled already by agreeing that we need both.

SIBLEY:

My intent was that there is a need for this large-scale data processing activity and the person who makes large engineering programs. This is one end of the spectrum. The other end is presumably the non-programming user. Between them there is an infinity if you like.

KIRBY:

How do you see the responsibilities of the applications programmer and the data base administrator with regard to the care and feeding of data? Does the data base administrator just have a responsibility to define the ground rules and then after that it is up to the people that are using the system to look after the quality of the data, or does his responsibility include such things as

scheduling the work?

SIBLEY:

I take an identical position to that of the controller of the company who has a staff of accountants. There is someone on his staff who is interested in the flow of money, and someone else who makes sure that the money flows. These are two different responsibilities. They happen to be vested in the same officer, but they are two functions. There are similar functions in the flow of data and information: somebody who makes the prior decisions based on the best knowledge he has, and someone that monitors and cares and feeds the data. These are both parts of the data administration office, but I would probably call the latter a systems administrator; he is what used to be the computer operator who has generally up-graded himself into being a manager. He, at the present time, schedules jobs and knows nothing much about anything else. In the future, he will become a much more responsible person (or this office will become an adjunct to the data administrator). This seems to be the fundamental duality of the data base administrator that we grapple with so much. On the one side he is a technician who has to understand exactly how the nuts and bolts are put together, and what duplications are going to be in the data base. On the other, he has the ongoing responsibility of caring for the data as a resource, (in which role he has to interact with both user and higher management).

Data Base Management Systems, D. A. Jardine (ed.).

LARGE SCALE DATA BASE SYSTEMS
CURRENT DEFICIENCIES & USER REQUIREMENTS

J. A. GOSDEN
The Equitable Life Assurance Society of the United States
New York, New York

1. SCOPE OF REMARKS

I have chosen to select eight topics for discussion of user data base requirements and each is a separate section of this paper. The following list of topics includes the year when I was first exposed to the topic or formulated my first concern about it and the references are to any available documents occurring, as in the case of the first topic, up to five years later. The purpose of showing these dates is to indicate the long time lags that occur in this field.

Incorrect and incomplete data - 1967 (Gosden (1972))
Data bases from data pools - 1968 (Gosden (1972))
Sturdy data bases - 1969
Distributed data bases - 1969
Mistress - 1970 (Gosden (1970))
Access control via construction rules - 1971 (Gosden (1972))
Common physical and logical boundries - 1972
Massive data base restructuring - 1973.

There are two topics for which I have pressed in the past that are not included in this paper. First, a range of different user languages suitable for different types of users (Gosden (1969), Gosden & Raichelson (1969)) because I believe that sufficient attention is being given to this by others, and progress is limited by pragmatic experience. Second, improved compatibility of data and related development of data description languages (DDL's) (Cosden (1969, 1969(a))) because that also is the subject of much study although, again, progress is slow due to the inherent complexities of the problem.

The topics that I have included need still more attention, and I hope to increase the developers' understanding of these needs, and to solicit support from other users who have the same problems.

2. BACKGROUND

In order to set this paper and its statement of requirements into perspective it is necessary to set out the author's background.

Although one major part of my experience has been in system software development, I have not been involved in the design or building of any data management system and am somewhat free of ownership bias. Another major part of my experience has been user oriented. A significant involvement, at least for the last ten years, has been with large scale data bases actively in operation in one form or another. Some previous examples include military command and control files, the MEDLARS bibliographic file, the DoD research reports systems, satellite data and most recently insurance data bases.

My biggest problems with large data base systems have revolved around the pragmatic problems of viability, including performance, convenience, reliability, soft failures and recovery.

My current concern is the Equitable's on-line data base of individual insurance policies. There are some 3,000,000 policies whose compacted form fills about 97% of twenty IBM 2302 disc devices. We are currently running our own system software on IBM 7080 computers with COMTEN 60 front ends to our network.

We are currently building a conversion system to be implemented under IMS on an IBM-370-168. The data base of some 2 billion characters will occupy about 26 spindles of the IBM 3330 disc units. During prime shift we will have about 100,000 transactions a day against the data base.

3. INCORRECT AND INCOMPLETE DATA

This topic is one of the important themes which I have discussed before (Gosden (1972, 1972(a))) and this section is a restatement. Although data management systems have moved far from their humble origins in the middle 1950's, their growth has been slow because it has taken a long time to recognize all of the problems related to automating the handling of data, let alone developing solutions to those problems. The biggest problem that has not been faced by developers is the pragmatic problem that data, unlike programs, can be plagued with errors, omissions and inconsistencies. Today editing, validating, vetting or data-scrubbing operations are usually provided by own-code or separate pre-processors. There are four exceptions, of which I am aware, two where general purpose input validators were built for military applications, the first for batch and the second for on-line use (Polis (1969)), third there is a commercially available package called "Editor" (Group Operations (1970)) and fourth another a package called "Editor" (Griffiss (1971)).

The amount of validation required is not just the simple set of validity functions such as control totals, hash totals, and some internal reasonable and consistency checks. Two extensions are required. First, the checks should be extendable to other files, both earlier generations of the same file and related versions of other files. Second, we would like the ability to annotate bad data. This is a facility to assist one in using bad data as opposed to having to wait for it to be corrected. We need three facilities:

1. a method of marking bad data with qualifiers to show missing items and to show which items failed which screening checks,
2. a method of specifying alternative or default replacement rules to apply when the original data is marked as bad,
3. a method of annotating data base files to indicate the less than perfect quality of items when alternative construction rules have been used.

Comprehensive alternative construction rules must be potentially elaborate, but even simple ones such as the following would be very useful: for a missing entry in a file use the value from the previous generation; for an illegal name (say LODNON) which is likely to be a typographical error, use it directly with annotation (e.g., LODNON?); or if accumulating values (by towns) put values associated with illegal names into a total called "Miscellaneous"; and for unreasonable data substitute a preset standard value. Most of the construction rules that are used today are implicit in people's minds and only a few programs have been developed that contain such rules. The difficult technical problem is to devise an effective way to describe such data relations and to provide a simple way to enable users and data specialists to use them. The user should not only be able to say what kind of data base structure he would like, but what he will accept instead. He should be able to specify detailed construction rules, alternatives, defaults and fillers, and also, whether he wants his fillers and qualifications annotated. If he is not adroit, he will have more annotation than data on his listing.

4. DATA BASES FROM DATA POOLS

This section is a precis of a form of data management that I have described at length elsewhere (Gosden, (1972, 1972 (a))). It is not possible to give more than an indication of the concept of constructing data bases from a pool in this section and the reader is referred to either of the cited references.

In essence, a data bank is proposed that contains two major kinds of data.

First it contains all the original source files in a data pool. For simplicity consider that these might be the on-going everyday data bases occurring in the execution of an organization's affairs. Second, we consider user's data bases which can be combined, modified, updated and manipulated for staff or administrative purposes, e.g., sales projections, market research, budgeting, auditing and product analysis. These data bases are kept separate from the pool and the activities of any user are constrained so that they affect no other user nor the pool.

5. STURDY DATA BASES

This paper reveals, among other things, that I am almost paranoid about defending (or protecting) my systems from "Murphy's Law". I believe that one takes an umbrella, not to keep off the rain, but to improve the chance that it will not rain.

One of the spectres that haunt me is that which is conjoured up whenever I listen to the priests of data base management systems describe marvelous visions of integrated data bases, multi-threaded lists and a plethora of pointers. I have nightmares of some errors causing the whole data base to unravel as when a loose thread is pulled in a knitted dress.

My requirement is important and simply stated, it is essential that we can create a large data base out of a collection of separate smaller sub-data bases. We want to be able to use the large data base as a single integrated logical entity, but when we have a problem we want to be able to localize its propagation, and be able to restrict the extent of any restart or recovery to only one of the smaller sub-data bases. In our use of several related data bases, we intend to exploit logical pointers rather than physical pointers to link sub-data bases together.

Another intriguing extension of this requirement is that I want to be able easily to construct my applications so that processing can continue in some only slightly degraded mode when a sub-data-base is unavailable, and also to continue while a sub-data-base is being recovered. This is a multi-dimensional problem. If, for example I have a data base that I can access by agent or policy-type, I want to be able to continue when either an agent subset or a policy-type subset is down.

I have not personally researched and set out the technical implications of, nor a technical solution for, this type of requirement in a precise form rather than the vague "arm waving" or "word weaving" used above; but my technical experts have assured me that current designs are hospitable to this requirement, and I want that type of requirement explicitly identified, retained and reinforced.

6. DISTRIBUTED DATA BASES

I believe that there will be a significant trend to distributed data bases (Summers (1967)) for many systems, whether for reasons of reliability, convenience, decentralization or economy. In fact if distributed data is feasible then the requirements in section 5, sturdy data bases, are also met. Indeed it may well be that the requirement for the sturdy data bases is only a stopgap until we can have distributed data bases.

Distributed data bases need one property in addition to sturdy data bases. This is the property of recovery and backup of some nodes in network by other nodes in the network.

7. MISTRESS

This section is an updated and more polished version of an internal document

(Gosden (1970)) that has been distributed in various data base circles since 1970, when it was embodies as a "desirable" requirement in a request for proposals for the CAPS system at Equitable.

7.1 Preamble

In a casual discussion in 1970, the author postulated a design concept which would be suitable for on-line systems in which the basic design of the control mechanism was oriented to testing, and in which operation was a special case. This has attractive advantages in that volume testing, field testing and cut-over can be done smoothly, and even very rapidly for emergency changes. This section outlines the concept. One major caveat is that our operating system may not be hospitable to this approach.

7.2 Grand Strategy

We assume a system implementation under our operating system. We develop a special controller (called Master or rather "Mistress" to show our benign concern) for on-line routines. As a basic simplification for discussion (which may have to be modified in full design), we assume that all on-line routines are developed as separate modules and that each module is entered from, and exits, to, Mistress. There are no modules which are sub-routines to other modules.

We assume Mistress is "transaction-driven" in that it progresses transactions independently through the modules as though each is a separate job in a multi-programming environment.

We must develop a scheme to segregate data (as would be needed for multi-programming or privacy) and the concept depends upon Mistress' ability to maintain such segregation. This is not unreasonable. Apparently, CAPS does this well today.

7.3 Testing Environment Scenario

We assume for simple discussion that CAPS exists and is running operationally. For our example we assume that an operational module "X" exists and a replacement module "TX" (Test X) is being tested. We assume that basic debugging of the module as a unit has been completed.

7.4 Data Base Regions

In the simplest case there are four data base regions. They are:

Operational Records
Master Test Records
Test Run Records
Test Results Records

The Operational Records are obvious. They are the live operational data. The others will be described in context.

7.5 Scenarios

We now describe typical processing of transactions, for different phases of testing up through cut-over. Mistress should have a parameter control mechanism so that variations in the phases can be imposed (see later discussion).

7.6 Operational Transactions

These transactions always use operational modules; they access and alter operational records. They may also do additional tests described below.

7.7 Type 1 - Test Transactions - Solo

These transactions do not read nor alter operational data. They use all operational modules except X, and instead of X use TX (Test X). The transaction uses master-test records only for first access but any output that would alter the data base is intercepted and written in test-run records. Subsequent access would be to test-run records. Thus at the end of a test, the test-run records can be cleared, and the master-test records are preserved. Note that no action is taken to affect control totals.

Mistress also intercepts all inputs and outputs to TX and copies these into test-results records, which can be listed at the end of a test.

As an extension, there can be many routines with test versions and the test transaction always uses a test-module if available. A further extension allows different "sets" of test modules to be established and a test-transaction is associated with a test set. In this last case separate listings would be printed for each test-set.

7.8 Type 2 - Test Transactions - Compare Test

For these types Mistress would execute both "X" and "TX" for the test transactions, and must ensure that identical input goes to both. Both sets of outputs are recorded in test results. The resulting listings could be any combination of

(a) TX input (=Xinput)
(b) X output
(c) TX output
(d) X-TX output differences

(a), (b) and (c) could be conditional on differences existing in (d). The most useful set of listings can be chosen later and varied with experience.

7.9 Type 3 - Test Transactions - Live Data Comparison

This is a condition on a module, let us call the module "CX" (Compare X). For such modules Mistress will for all, or some sampling, of live transactions route them through X, update operational records and copy results into test-results. Then it enters the same input to "CX" and puts the output into text-results. The listing can be the same as for Type 2.

At this stage we see the need to limit the volume produced on test-results and for operators to be able to turn off testing if system performance is being degraded.

In this case the output of test-results could be limited to differences.

7.10 Type 4 - Acceptance

This is a condition on a module, let us call it "AX". For such modules we invert the procedures of Type 3. The "AX" output is used as operational and "X" output has a comparison. In this case only differences are recorded in test results.

7.11 Type 5 - Probation

This is the same as type 4 except that no comparison is performed. The old X sits there quiescant.

7.12 Some Comments

In both types 4 and 5 there is an emergency operator option to "revert" to "X" if problems arise. Reverting is probably best controlled by only introducing a batch of Types 4 and 5 each week. Then if a crisis arises during a week that a restart does not cure, then a restart reverting to the previous week's version of modules can be attempted.

In practice Mistress should be designed to have minimum overhead for cases where there are no special modules. However the benefits of such a system are probably worth a significant overhead.

Mistress can be parameterized to allow different numbers and kinds of types of testing.

Provided restart is also transaction-driven we should have few problems in over-all conttrol.

Note that such techniques could be applied to batch processing but would involve a dramatic change to program architectural approaches which should not be considered until this has proven itself in on-line design.

This approach avoids the need to maintain a test version of the programs and allows new modules to be tested under load conditions, or even excess load by combining operational plus test transactions.

8. ACCESS RIGHTS

Most schemes to automate the access rights of users to data have been variations of assigning classification level and need-to-know codes to groupings of data and users. This approach becomes very difficult to administer in large organizations which then tend to the solution of clearing everyone to the highest level. This approach also fails to deal, directly, with the concept that it is particular collections and correlations of data items that need to be protected, not the data items themselves.

Serendipitously, the data-bases constructed from a data-pool concept mentioned above have a structure whereby user's data bases are defined by the rules for combining data. If then the security officer must approve these rules for a set of users he is given a powerful tool to make it easy for him to see the aggregations and potential correlations that users could exploit.

For example, most companies would make the item "wage-rate" or "annual salary" subject to tight restrictions, whereas the important point is to restrict the person and salary association. Thus, it might be reasonable to allow wider access to salary and grade associations for certain purposes. Conventional schemes are clumsy at allowing this.

9. COMMON PHYSICAL AND LOGICAL BOUNDARIES

Sometimes it seems to take a long time to retrieve some useful second generation features in the third generation. A well known example is the regaining of hands-on console debugging by the use of time-sharing facilities, which still have not completely dealt with the problem of large scale commercial systems.

An interesting example exists in the relationship of physical and logical boundaries in a data base. CAPS, the large comprehensive individual insurance on-line system of The Equitable still uses some twenty IBM 2302 second generation disc devices to hold records for some 3,000,000 policy holders. For various historical reasons there are strong correlations between the logical and physical data arrangements. For example the records are divided into 10 logical subsets and each subset is allocated to a pair of disc devices. The subsets are recognized by the value of the least significant (terminal) digit in the policy-number i.e., policy 12345678 is in subset 8. Other relationships exist, e.g., the system keeps accounting control totals by cylinder, and it records hard errors by individual tracks.

The system can continue running while a subset is "down", and can continue running while individual tracks are "down", either while maintaining accounting controls. When a subset is down a user of the system can also react intelligently for example we can send out a broadcast message that terminal-digit-7 is not being processed.

Thus CAPS is a sturdy data base (see section 5).

The alignments between logical and physical boundaries allow us to "take down" a physical part of the system because of malfunction, without taking down the whole system. Thus a physical error can be confined to one logical area, and, vice versa, an error in a logical area can be isolated by a hardware switch. Thus if we have to replace a component or make a recovery after a repair to a major component we can limit the operational impact to a logical area of our data bases.

Once again notice how distributed data processing should automatically meet this requirement.

Let me reinforce this requirements with more illustrative detail. When we get hard errors on a track we "turn it down" and set an appropriate entry in a table. If we get transactions that require access to that track we return a message "record not available" to the application program. This causes a similar message to be sent to the on-line inquirer and a report to a control function which usually re-initiates the transaction when the problem is resolved. This is a suitable technique for infrequent events. But if we have a problem with an entire disc unit that takes down one-tenth of our data-base we would have 10,000 of these events per day. In such cases we can, with one broadcast message, inform

our users that "all records for terminal digit 5" are unavailable for processing, and trap the related transactions without flooding the system with warning messages for each case. Without this type of logical large-scale lock-out we would probably have to shut down the entire system since we could not identify the class of transactions that would be affected.

10. MASSIVE FILE RESTRUCTURING

One of the advantages of "data independence" is that there are opportunities to restructure the data base, either to insert and delete items, or to make rearrangements for improved performance. While these are logically simple to make, they are still very expensive, take a considerable time and could be disruptive to schedules.

Let us consider the CAPS file, a large, but not enormous file of some two billion characters. This will use about three fully configured 3330 disc units, or 26 spindles.

Our current estimates show that a straightforward rearrangement would cost, at our internal rates for computer usage, some $60,000. This is equivalent to 32 hours of fully loaded revenue. We estimate that the elapsed time may well be 72 hours or 3 days.

Before we can really evaluate this amount we need to know how often it may occur. Even with our second generation system we have made two rearrangements in the last year to tune-up the system. On those occasions we have had to plan the rearrangements for long holiday week-ends. As we convert to third generation we anticipated the need, and logically planned to be able, to make several tune-up rearrangements. Thus, the estimates I quotes above were not reassuring.

It is not as bad as it looks at first sight. Although CAPS has been considered as one large file, it has always been managed in ten separate parts, one for each terminal digit (see section 9). We have decided to continue this practise for the reasons explained in the previous section; i.e. robustness of the data base, thereby reducing, and logically controlling, problems that seriously affect the data base. But it would still take one full day on each of three week-ends to rearrange the CAPS file.

Now it turns out that under IMS we happen to be able to divide our major data base meatly into ten subordinate data bases. Let me repeat: as it happens. Our requirement is that it be a deliberate explicit facility, not a happenstance, not a facility that we achieve by some devious means.

Let me summarize the requirement. We want to be able to deal with a super data base that is composed of separate sub-data bases with differing physical and/or logical file structures. We want internally for the application programs to be able to deal either with the super-data base as a whole, or the sub-data bases as separate entities. We want to be able, externally, without disruption to, and in coordination with, the internal applications, to initiate, restart, rearrange, or temporarily freeze a sub-data base.

11. SYNTHESIS

It does not take long to begin to realize that four of the topics that I have covered:

- sturdy data bases
- distributed data bases
- common physical and logical boundaries, and
- massive file restructuring

tend to be achieved naturally if we achieve their glamorous member distributed data bases (DDB).

DDB automatically makes it possible to have, and control, common physical and logical boundaries, which in turn can be a form of sturdy data bases if any interconnections are restricted to logical links and pointers, then, in addition, if each DDB has its own local physical structure we can divide up the problem of

file restructuring.

One is tempted to state his composite requirement as being a request for a "virtual distributed data base".

DISCUSSION

BASH:

You discussed techniques for soft fail. It looks like IMS takes the other point of view. It is absolutely unforgiving if you have a damaged data base. Are you planning to change this?

GOSDEN:

First we are going to put up a pilot version on IMS. Then we are going to compare the pilot with the 7080 system and then decide how much improvement we need before we cut over. In the meantime we are trying to exploit all the legal facilities available such as creating 10 different data bases and using a Data Interface Program to resolve the problems. We have a plan to stay on the 7080 until 1976 if necessary. We just think that that is the most cautious way to go. Four years ago we were told that we couldn't put all of CAPS on the 7080's, but we will, and with a company like ours you find a way.

TURNER:

One of the concepts behind a data base system is a shift from the domain of the applications program into the domain of the data base management system. Whether this is practical or not depends upon how much of the total system is involved in things like recovery and similar support functions. What portions are made the main line functions and what portions are the support areas?

GOSDEN:

All the major requirements that I have discussed are in some DBMS area, not in applications.

TURNER:

Are you planning any organizational changes or have there been organizational changes?

GOSDEN:

Yes, we are slowly reorganizing. The second generation system has grown its own data owner, data administrator, and maintenance facilities, Unfortunately there isn't just one place in a big organization where everything to do with data bases can be centralized.

TURNER:

Could you give us the reference to the paper by Vanevar Bush?

GOSDEN:

No but I will put it in the paper. It is the Atlantic Monthly in the late forties or early fifties. Your librarian really should know it.

JARDINE:

One of the things that I have consistently heard for the past two days is the

problem of reliability of the data base. This usually elicits the question from vendors and programmers, how much are you willing to pay? What's its value to you? Those are not proper questions. The message I get is that people are already paying a great deal for reliability. They are paying for it in duplication of data, redundancy, the amount of systems programming effort that has gone into recovery, fail safe procedures, dumping data bases, etc. There is a very loud and clear message to the software and hardware manufacturers that reliability is now costing a great deal of money that is not visible. I would hope that at the end of this meeting we might get a very clear idea that people want reliability and it isn't a question of 'what are you willing to pay for it' but that you can't run without it.

GOSDEN:

We pay a lot today...we would pay more if necessary to stay in business.

BACHMAN:

I would like to make one or two comments about the reliability issue. I happened to be sitting in a conference in Florida four years ago at the time of the first Apollo launch that took people successfully to the moon, landed, and brought them back. It turns out at that time there were also some ex-Apollo types that were meeting with us and we were marvelling about this perfect flight to the moon and back. The Apollo types said, 'Well, let me stop you for a moment. I am sure there is a list as long as your arm and my arm put together of all the things that went wrong in that flight'. But they had a very careful set of priorities in terms of what they should do first. The highest priority was to bring those men back alive. The second priority was to get them to the moon and back. The third priority was to get them down onto the moon. They had a good understanding of what they had to do in each case. I think in our computer role we have to look at what our strategies are. For our purposes I have established three priority areas: the data base, the communications and job exectuion. The highest one is the data base. The reason for this is that it is the toughest thing to repair if you lose it. The second priority is to communications. If you lose something you hate to go back to the person at the terminal in your various branch offices and say, 'Please retransmit what you sent a half hour ago because I lost it', although that is less embarrassing than saying, 'Please recreate the files that you submitted five years ago'. The least priority is aborting a job that you started. Nobody outside knows about it; it is a part of the overhead of running the business. You have to rank these things very carefully in planning all these systems. There is one more thing that you have to worry about, which goes back to a rather humorous story that I might tell for a moment. I was visiting a railroad company on the west coast, and I had read in a news magazine that railroads were losing box cars and so I brightly said, 'I understand you are losing box cars'. The customer said, 'No, no, I have never lost a box car, but we occasionally lose information about where they are'. So certainly the highest priority is retaining system information about where the data base information is.

Data Base Management Systems, D. A. Jardine (ed.).

DATA MANAGEMENT SYSTEM REQUIREMENTS

J. D. JOYCE, J. T. MURRAY, M. R. WARD
Research Laboratories
General Motors Corporation
Warren, Michigan

INTRODUCTION

The purpose of this paper is to give a summary of user requirements for data management systems as observed from some of the operations in a large manufacturing business. To put the observations in perspective, the first portion of the paper will give the flavor and characteristics of a few applications of data management systems within a large manufacturing company. These observations result from discussions with users who have expressed their concern over the limitations of current data base management systems. We have attempted to classify user needs into short- and long-range categories. Undoubtedly there is some overlap. By short-range we mean a period of less than five years. Solutions to the short-range needs may not require new technology, but may simply be a matter of bringing known technology into productive use. The long-range user requirements focus on the directions of development of data management systems which will bring the most benefits to the users.

Long-range user requirements in data bases will influence both the software and hardware developments that emerge. Likewise, the trends of computer architecture toward distributed systems, and the changing technology of larger, faster and cheaper memories of all classes will tend to influence the way in which these user needs are satisfied. Some of these influences will be described in the last section of this paper.

One common term used in this paper is data base manager. This term refers to the package of software which interfaces with and controls the data base which usually resides on secondary storage. Data administrator refers to a person who is responsible for the policies and operations of a data base.

DATA BASES IN A MANUFACTURING BUSINESS

In order to describe how data base management systems are used in a large manufacturing business, it is useful to examine the process of developing a product from the initial design through the end of the manufacturing cycle. For every product there is a model number, for every model a list of its major assemblies, and for every assembly there is a list of subassemblies, etc. It is the responsibility of the various engineering departments within a manufacturing business to design the parts for a given product. The rest of the manufacturing cycle is then concerned with producing and delivering the product.

The design departments develop the initial designs. Purchasing and receiving are concerned about getting in the raw materials from which the product will be made. Processing pushes these materials through the various stages of production required. Shipping, sales and service distribute the product to the consumer. From this point of view, a major data base of a manufacturing business is the product structure file describing what is made from what. Hung off this master data base at various levels in the structure are data related to receiving raw materials and manufacturing, delivering and servicing the product.

As a hypothetical example consider the data base diagram, Figure 1. This diagram illustrates some of the information that may be required for each part, subassembly, and assembly in the product structure file. For a given part number there will be design data, engineering specifications, inventory, and warehousing

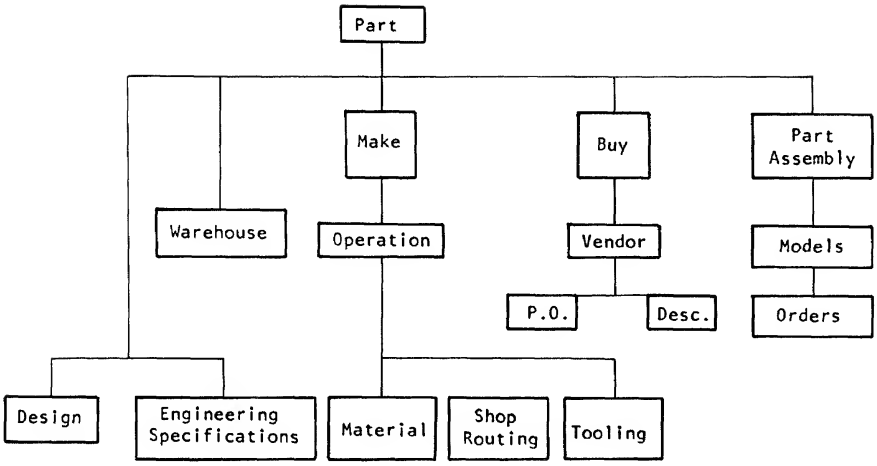


FIGURE 1

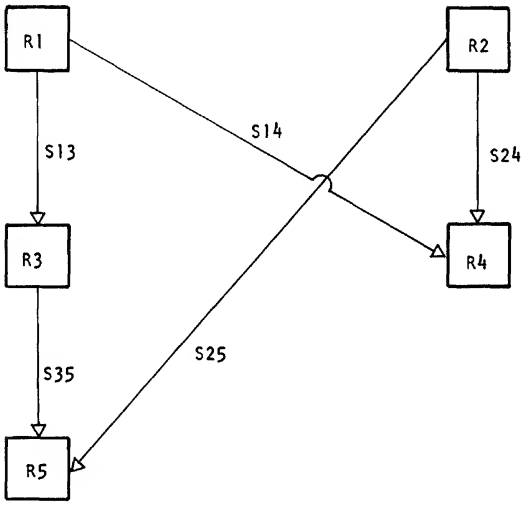


FIGURE 2

information. Inventory and warehousing data include quantity on hand and location of storage. This information is updated when new shipments are received, when parts are moved to assembly lines, and when spare parts are shipped to purchasers. There will be detailed assembly (parts explosion) information which describes all component parts or subassemblies for this particular assembly. The minimum information kept is the part number and the quantity used for each subassembly. The inverse of this information on detailed assembly answers the questions: Where is this part used? What assemblies does this part go into? What models are affected? What current orders are affected by this particular part or assembly?

Although no totally integrated data base exists for any major manufactured product, some specific examples of current data base applications will be described. Each of these systems would make up one portion of the hypothetical system just described. The variance within these existing applications highlights the difficulty of generating a fully integrated data base system to handle all of a manufacturer's data base needs. The applications are:

1. automobile body design,
2. engineering specifications release,
3. material flow control.

The automobile body design application involves the use of various mathematical procedures (line and surface smoothing, finite element analysis, etc.) to develop and evaluate potential designs. An elaborate data structure is required to represent the geometric relationships between the points, lines and surfaces in the mathematical model of the proposed design. Flexible access to the elements of this data structure is obtained using a data base management system developed in-house for this purpose. (Dodd, (1966)). While the data bases for this application are small by commercial standards (on the order of one million bytes per design), the relationships between elements are considerably more complex. Over half the storage requirements of the data base are for data pointers. Furthermore, using graphic consoles with line drawing and light pen facilities, designers can impose very heavy computational loads requiring continual access to the data base.

Another application of data base technology in the manufacturing industry is the continual process of releasing new or revised engineering specifications. This application focuses on the official releasing of documents describing parts from engineering status to production status. Important adjuncts to this application are maintenance of master parts lists, paper work to assembly plants, and purchasing of parts. To give some perspective on the above application, the approximate size of a current data base in one installation is 250×10^6 bytes.

Approximately 40 terminals access this data base and generate about 5000 transactions per day. These indicators of data base magnitude and activity are not intended to be thought of as typical. There are a great diversity of data base applications.

A number of data base applications contain information describing the acquisition of parts. Any particular part may be made and/or bought by a manufacturing plant. If it is made, then detailed information on the fabricating operation is needed. Included may be further information about the material required to build the part. Shop routing information is required which describes the way the part flows through its assembly operation. Finally, tooling information is required which indicates required tools, amount of labour, set-up time and rate of assembly, etc. All of this information is required for shop scheduling. If the part is purchased from a vendor, then information about all vendors supplying the part is needed, as well as lists of outstanding purchase orders.

After a part is acquired, either through manufacture or from a vendor, then another phase of parts control is entered. Important existing applications of data base systems are material control, inventory control and warehousing systems. Some applications use commercially available data base management systems and others use specially designed in-house systems. The basic function of these systems is to monitor and control the flow of material into, within and out of a manufacturing plant or division. When a part arrives at a receiving dock, an

entry is made into the data base as to quantity received, where it is to be stored, etc. At some later time an assembly line will have need for some number of parts. After it is determined which one of several warehouses a part is to come from, a move order to transfer the parts to the assembly location will be issued. When the parts are actually transferred, the data base is updated from the warehouse to reflect the change.

Subsequent transfers of this part from station to station within the assembly process may also be recorded in the data base. In addition to monitoring part movement these systems are used for determining when to reorder parts and whether sufficient parts are available for scheduled production. A warehouse system may be set up for spare parts and includes order entry, filling of orders and shipment. Parts inventory systems may be used for nonproductive items such as screwdrivers and wrenches as well as production parts.

The product structure file which is partially described above contains only the data relating to the manufacture of a finished product. This file tells which operations on which parts are required to make the product. Nonetheless this file is required, along with additional raw data on sales, for applications such as forecasting, transportation logistics, finished product inventories, or overall production scheduling.

What are some of the characteristics of existing data bases? Documentation and user interfaces are essential for routine operations. In complex structures, dictionaries are needed which define abbreviations, contain text for reports and CRT displays, and contain an indication of whether quantities are in terms of square feet, rolls, hundreds, centimeters, hours, etc.

Existing systems have as few as one terminal and as many as 100 terminals. They may be inquiry only, with batch update or may include both online inquiry and update. Existing data bases range in size from a few hundred thousand bytes to a few hundred million bytes, with up to one-half million part numbers. Transaction rates may be as high as 10,000 per day from a 40 terminal system with as many as one-half of these transactions being updates to the data base.

A number of the data base characteristics for these material flow applications are established by the nature of the product itself. One characteristic is the size of the data base required. Simple standard assemblies involving a few dozen types of parts can be managed with a few hundred thousand characters of data base storage, even in high-volume production. The larger products with long service lives may require record keeping on hundreds of thousands of different parts with parts explosions reaching to 16 levels. These applications require data bases with several hundred million characters of data base storage, even in low-volume production.

Another characteristic is the rate at which transactions are entered against the data base. High-volume applications involving highly standardized parts can be monitored on a pallet-by-pallet basis requiring perhaps only a single teletype at the shipping and receiving dock. Low-volume products with complex order specifications and long lead time are more difficult to keep track of and require scores of terminals in shipping and receiving, engineering, assembly, sales and accounting.

All these applications are sensitive to material flow problems such as missing or incorrect parts. High-volume applications are hurt by these problems because of the expensive machinery that is idled. Low-volume applications with large products are hurt by the difficulties and expense involved in maintaining the unfinished product on inventory while awaiting a missing part. Thus reliable performance from the data base system is essential.

Some problems are unique to each of the applications. In the high-volume subassembly operations, there is the problem of interfacing the separate data bases involved. Changes in demand for a high-volume product may be monitored at a central office, and these changes must be quickly disseminated among all the subassembly plants.

A significant part of a low-volume, long-life product is the renovation of existing products. This requires maintaining the already large data bases for long periods of time.

It is important to note that our discussion of data base applications has

dealt with fragmented pieces of the total picture. We do not know of a system which handles the data described above in a completely integrated fashion. Part of the reason for this is that data base technology is just now reaching the point which such a complex system can be considered practical. Another factor slowing the development of these integrated systems has been the usual decentralization of manufacturing activities in most companies. For example, GM makes automobiles; but in most of GM's manufacturing installations the product is a radiator or disk brake assembly, transmission or automobile body. Within each manufacturing installation the product is just one stage in the production of a radiator, transmission, etc. Thus instead of one large, integrated data base there may exist separate data bases at various installations concerned with various subsets of the overall problem. There is no master data base describing all the raw materials on order at any point in time. But there are a number of material flow systems at specific installations to monitor and control the flow of material into, within and out of a plant. There is no master product structure file describing all the engineering data for all the parts that go into all the products of a large manufacturer. There are individual files of engineering change and release notices. Applications of data base systems thus include a spectrum of product-related applications involving the receipt of raw materials, manufacture, sales, and service of manufactured products.

Having given a flavor of some of the data base applications in a manufacturing business, we move on to the user requirements for improved data base technology and use of this technology. We believe that the range of data base applications in a manufacturing business and the requirements are not significantly different from what we would expect from commercial, high technology or other general business areas.

SHORT-RANGE USER REQUIREMENTS

A number of user requirements which should be met within the next few years will be discussed in this section. These requirements include the following:

1. User education
   a) Data administrator
   b) Use of data base systems
2. Online access
3. Concurrency of updates and retrievals
4. Graphics
5. Independence of data description and programs
6. More flexible and powerful data structures
7. Performance
8. Data base design guidelines
9. PL/I interfaces
10. Common data base languages
11. Self-contained query languages

User Education

Education of users is a prime requirement to make effective use of data bases. This single topic could cover many aspects of data bases. Only two items will be singled out here for comment. Before large integrated data bases are assembled, a data administrator should be appointed with sufficient authority to take on responsibility for the integrated data base. This responsibility should encompass more than just the technical maintenance of the data base. The data administrator should have enough authority to mediate or decide on courses of action when departments of a company have conflicting needs. The educational problem referred to is that of educating users to clearly define the areas of responsibility and authority for a data administrator. The data administrator should be appointed at the beginning of a data base project. In addition to playing a key role in managerial types of problems involving interdepartmental mediation, policy making, data security controls, etc., a data administrator and his staff will be highly involved in some of the more technical areas. Data base

design requires a good understanding of the nature and amount of load each application puts against a data base. Applied knowledge of the trade-offs in design and of the performance characteristics is essential for successful data base operations. The data administrator has the opportunity to standardize data base terminology across applications and thus be effective in reducing confusing documentation and ensuing maintenance costs. There are many other aspects of this data administrator function which cannot be addressed here, but the importance of this function is too frequently downgraded.

Another educational problem is that of teaching users what to expect from a data base system. It has been observed in many instances that users, even online users, will generate requests which produce large stacks of standard reports just as they did when working with older systems. The users need to be educated to getting exception types of reports, or condensed versions of reports. Online users need to learn to interrogate the system for the information they want rather than using the online terminal as a trigger for large reports.

Online Access

One of the user requirements of data base systems in the manufacturing industry is the need for online access to a data base through many varieties of terminals. A data base manager with only batch capability is gradually being phased out. Terminals are either planned or already located on shipping docks, receiving docks, in assembly plants, in warehouses, in product dealerships, in engineering design offices, etc. These terminals are being supported by teleprocessing packages which are either an integral part of a data base manager or are interfaced to a data base manager. Although these teleprocessing packages greatly ease the user's effort for program development, the variety of terminals (each of which are always a "little" different) still causes considerable problems.

Concurrency of Update and Retrieval Operations

As online terminals become the predominant interface to a data base manager, additional capabilities are generally required to support concurrent operations. If the online terminals are used only for retrieval, the required concurrency of operations is fairly easily accomplished. The supporting system programs are made reentrant or are simply replicated as required. However, as the terminals become used more and more for online update, much more extensive changes must be made to the data base system to support concurrent operations. For example, with concurrent update, the log tape of changes made to a data base reflects alternate changes made by several users concurrently. If one of the users gets into trouble, the rollback procedure must be carefully written to restore only the records he changed and no other. Furthermore, a fairly sophisticated interlock facility must be established to ensure that rolling back the changes made by one user does not affect subsequent changes to the same records made by another terminal user.

In many cases data base management systems are simply not designed to support concurrent update operations; users forego applications requiring this capability. In other cases partial solutions to the concurrency problem are devised which work only with carefully written application programs. Application programmers should be freed of the necessity of resolving update conflicts, and users should be spared long delayed responses because of inadequate provisions for concurrency of updates. The concurrent update problem is solvable and has been solved in a few systems; better solutions are needed very soon in the remaining data base managers.

Graphics

An example of online access worth special mention is graphic consoles. Some of our applications, particularly those with an emphasis on analysis of data, demonstrate a growing need for graphics, both passive output and interactive graphical input/output. Histograms, financial bar charts, correlation plots are examples of passive output, which may be displayed on interactive terminals. Interactive engineering body design using highly interactive terminals is an

increasingly important application involving a large data base consisting of mathematical data representing the points, lines and surfaces of an automobile section. Structural and tolerance analysis using similar mathematical data are other applications requiring graphic capability.

This increasing use of graphic consoles has some important implications on data base systems: First, a sophisticated user at a good graphic console can use computer resources at a rate many times greater than that of simple inquiries from a typewriter terminal. This new user poses many new problems in areas of data base design, performance, and the data base managers' facilities and capabilities. Second, greater facilities must exist for interfacing new terminals, both graphical and non-graphical, with existing data base managers in order to handle requests for these specialized types of input and output.

Independence of Data Description and Programs

One of our users' needs is that of separating the description of data from the programs developed to process this data. A library of application programs developed over several years is a formidable thing to change. At present, however, one cannot change the structure behind a data base without affecting existing programs. In practice, what this means in many manufacturing installations is that new demands on a data base are warped so as to have minimal impact on old application programs. For example, an installation may have a large library of programs written in assembly language, yet wish to do future programming in a higher level language. The application programmer may encounter considerable difficulty in describing in the high level language a data format which was easily described in assembly language. One should not have to write clumsy code in the high level language to circumvent this problem nor should he have to change the existing data base and application library. The description of the data base should be separate from the application programs in such a way that one description serves programs written in different languages. Furthermore, if it becomes necessary to add new data fields to records it should only be necessary to change this common data description, not the existing assembly and higher level language programs in the application library.

This area of users' needs is expected to be a long-range need as well as a short-range need. Improvements will gradually evolve over many years. There will not be a sudden breakthrough to solve this problem.

Flexible and Powerful Data Structures

Another important user requirement is for more flexible and powerful data structures. Most existing data base systems require a static definition of the logical data relationships and do not provide dynamic (at execution time) capability to extend these relationships. Two examples of this dynamic extension are the ability to allow a record to be a member of an arbitrary number of sets and the ability to define sets of data to an arbitrary nesting depth (recursion). Both of these features require execution time support in the data base system which allows easy user programming for traversing up and down these sets with all the proper status information being maintained by the data base system and the host language. An important problem is providing these capabilities in an efficient implementation. Existing sequential, indexed sequential and random data structures do not provide sufficient capabilities to satisfy these requirements. New means of defining set relationships may have to be defined before efficient use of such capabilities is possible.

Performance

The performance of a data base system is of vital concern to the users. In making the transition from serial tape file systems to a more elegant data base system, a number of users have experienced very long running times. This situation has happened often enough that users are reluctant to put in improved capabilities unless good performance can be reasonably demonstrated first. Performance of online systems, of course, is another concern. Here the bottlenecks may be the teleprocessing package, the data base manager or overloaded hardware.

The requirements for good performance from a system with several potential bottlenecks naturally lead to a requirement to measure and predict performance. Is response slow because the teleprocessing package is not polling the terminals frequently enough? Or is it polling too frequently? Or is the data base manager slowing down trying to maintain the journal tape? What if we change the polling rate, or get a faster computer, or use more core, or get a faster drive for the journal tape? Can the system support a dozen more terminals using a new data base? What if these terminals require frequent access to the old data base in competition with existing applications? What if the terminals update the old data base? These are just a few examples of why users need to be able to predict running times for batch jobs and response times for online systems before they are installed.

How does one objectively measure and evaluate data base manager packages? Most operating divisions of manufacturing companies are users or potential users of data base systems produced by software or hardware vendors. In many cases, operating divisions cannot design and develop their own data base management system. Their programming resources must be dedicated to developing their application programs. Consequently, a good quantitative method of objectively measuring the performance of various features of a data base manager would be a useful aid as part of selecting a data base manager. It is undoubtedly true that for any data base manager, there are a wide variety of parameters. Thus any performance tests must be carefully controlled if any sort of reasonable comparisons are to be made from numerical results. Given that performance measurements can be reasonably made, extrapolations must be made from benchmark performance tests to estimated performance of the user application programs.

In an attempt to quantize the performance aspects of data base managers, GM has defined a set of transactions to be run against a given data base and timed. This method is referred to as the kernel approach. The logical relationships among records are specified for the data base. A 60 million byte population for the data base is obtained from an existing application by blanking out all but the numeric keys. Over thirty different types of transactions to be run against the data base are described and hundreds of samples of each are generated to produce statistically reliable test results. The logical data base structure is shown in schematic form in Figure 2. The boxes represent record types labelled as R1, R2, R3, R4 and R5. The directed lines between record types indicate sets in which the owner of a set is the record type at the tail of the arrow and the member record type occurs at the head of the arrow. The labels on the arrows indicate the names of the set types. There are many occurrences of each record type and set type. Included among the variety of transactions are retrievals based on keys or ownership or membership of sets, additions of records in ordered sets, modifications to records, changes to set relationships, deletions of records based on several criteria, and sequential record processing. The transactions are run in a specified order against the data base. Additionally, failures are simulated followed by restart procedures. Each transaction is measured with respect to elapsed time, CPU time, and number of I/O requests. The results are made available to interested GM groups without recommendations. It is expected that those groups who are interested will apply their own weighting factors to each of the kernels of concern to them. Then they will come to their own conclusions about performance relative to their own application.

Two main points are brought out by the measurement efforts described above. First, no measurement techniques or packages are commercially available to quantitatively evaluate the performance of data base managers. Second, performance evaluation is a very complex task, requiring careful controls on the tests and environment. Development of good techniques requires further effort.

Many users of data base systems need much better performance monitoring tools. Many of our users devote a very large percentage of their computing resources to supporting data base applications. Knowledge of performance bottlenecks can often result in dramatic improvements with simple changes. Anyone associated with data base technology has heard several horror stories of startling performance changes. While some vendors have made serious efforts to provide performance statistics, our users feel they need better tools which are

easier to use for most packages.

Data Base Design Guidelines

Another area of acute need for the users is that of more rigorous techniques and guidelines in the design of data base structures. Bitter experience by many users has shown that a poor data base design which does not match their application needs can have nearly disastrous results. Although the users emphatically express their concerns about data base design techniques, their requirements could probably be better met by much more program and data independence. If one could change data base definitions more easily, then getting an optimum design at the beginning of an application project would not be so critical.

PL/I Interfaces

For many years, COBOL has been the accepted high level programming language for the commercial programmers. In one manufacturing company more than half of the programmers on commercial applications are now using PL/I. A number of data base managers can now interface to both PL/I and COBOL. Thus, users need this interface to PL/I or in many cases a data base manager must be rejected without further consideration.

Common Data Base Languages

A common data base language that is easy and convenient to use and that has a reasonable degree of flexibility would be of considerable help to our users. At present, a language proposed by CODASYL is the only one that comes close to meeting such requirements. (CODASYL DBTG (1971)) Adoption of a common language would allow application programs to be written which would have some degree of transferability from one implementation of a data base manage to another. This would give more flexibility in switching either implementations of data base managers or even of hardware vendors. This would also mean that application programmers trained on one set of hardware and data base manager could work effectively on another vendor's hardware without the retraining. In some installations, multiple vendor's hardware supporting data base applications run side by side with a common set of supporting personnel. More readable source code and a reduction of the amount of source code to program applications would reduce the cost of development, modification and maintenance of applications. While the CODASYL data base language proposal has a number of deficiencies, it would provide a better base from which to switch to some improved level in a few years. Use of a common data base language would also provide user feedback on what would be desirable features in either a new or improved language.

Self-contained Query Languages

Better query languages are needed for the end users of data bases. Query languages should completely avoid the languages of the computer expert, e.g. job control language, FORTRAN, COBOL, PL/I, etc. Additionally, these query languages should be expressible in terms that are applicable or tailored to the user's work. Better query languages will help the users realize that they can get useful information out of a data base without piles of sequential output.

Many of the items discussed above are within current technology. It does, however, point out the gap that exists between the current technological level and that which is properly packaged into useful form for users.

LONG-RANGE USER REQUIREMENTS

One of the growing needs of the data base users is that of supporting unanticipated queries easily and efficiently against large data bases. This need is already evident in at least two areas. The first area is that of market analysis. The second area is that of product failure analysis.

In the first area, many variables in product demand need to be analyzed in unanticipated combinations. These analyses help in the planning decisions for production mixes in order to build what consumers are most likely to buy. In the

second area, feedback on failures of particular parts are filed in computerized data bases. Pre-planned extraction of information in standard report forms may help in determining correlations of causes of certain types of failures. However, for some time the need has been recognized for a capability to ask unanticipated questions of the data base which will give clues as to failure causes as soon as possible after the data are available. Obvious extensions are to ask logical combinations of questions.

The main point of the foregoing examples is to indicate the need for a capability to answer unanticipated questions. One may then ask if the above two types of applications are special cases and therefore do not apply to overall data base needs. The need exists to a lesser degree in many other applications of data bases. This may be expected to increase. Many applications have only recently used integrated data base managers. It takes a large investment in software, manpower and elapsed time to move through more advanced levels of data base technology. After the benefits from integrating data bases for routine repetitive operations have been well established, more flexible inquiries will continually be demanded of these data bases.

What techniques are emerging to meet these needs? Clearly, the trend is away from the undue concentration on processing individual records of information. Manipulation of sets of information is the predominant theme of the more advanced applications. A key aspect of this new emphasis on manipulation of sets of information is that new sets of information are dynamically produced and can be further processed. This latter feature is a significant departure from most of the current level of data base technology in which the relationships among records are predefined by specified sets and it is normally very difficult to dynamically add new set relationships or modify existing set relationships. Terminology to describe this emerging technology varies from simple terms which pertain to particular implementations to more esoteric theoretical terms. One implementation of representing sets of information can be described in terms of tabular sets of rectangular arrays of rows and columns of information. Other terms which describe similar concepts are relational data bases and set theoretical data bases. The term "normal form" is also used for describing how data relationships can be expressed. Terms such as "domains" and "n-tuples" are used where, in the context of a simple rectangular matrix, domain refers to a column and n-tuple refers to a row.

The implementation of such techniques within GM (Whitney RDMS (1972)) has taken a form in which each domain represents an attribute of the items being described. Each n-tuple represents a particular instance or occurrence of a group of attributes which pertain to a particular item.

Although implementations now exist which provide quite an extensive range of set manipulation capabilities, we believe it will be some time before such technology becomes commonly used on large data bases. One of the stumbling blocks is the failure of ordinary users to adapt to thinking of their data as made up of sets and of data base manipulation as set operations. Additionally, the importance of good user interfaces is frequently overlooked in data base management systems. Another problem is the concern over efficiency of set operations against large data bases. While some delays are tolerable when asking unanticipated queries, such delays are usually deemed unacceptable for simple pre-planned queries. Structures specifically built for pre-planned queries can normally be quite efficient, particularly if properly tuned.

To a user, efficient performance and good cost/benefit ratios for new capabilities can be obtained through clever software techniques or can result from new types of hardware. One of the avenues of new hardware techniques that has appeared promising for many years is the use of associative processing techniques to effectively search large amounts of information in a parallel fashion. All of the hardware that has been brought to our attention thus far has failed in this regard either now or apparently in the near future. Most hardware with associative memories have been small experimental packages not suited for commercial use. Those associative memories which are now emerging are very expensive and still of a modest size. While the modest size of an associative memory may not be a complete deterrent in handling large data bases,

good organization of I/O devices for sustained high bandwidth into an associative memory have yet to be developed. We anticipate that continued declines in costs of circuitry for associative memories and processors will bring this technology into useful data base experimentation. Eventually associative processing will become an integral part of data base technology.

The use of specialized hardware to process searches of large data bases will most likely be seen in the form of specialized data base computers. This will be part of the general trend of computer technology in which computers with either specialized hardware or specialized software will serve particular areas. The trend is already evident with specialized network or communications processors, processors for intelligent terminals, more intelligent I/O controllers, and, in some cases, processors which take the place of conventional I/O controllers. Specialized data base computers will emerge to carry out sophisticated data base modifications, searches, additions, range checking, validity checking, and security checking. What is the connection between these trends and the long-range user requirements? These architectural trends basically mean that a higher performance for data base operations over even larger data bases will be made available to meet user demands.

Data encryption will eventually emerge as a user requirement. Several factors are going to bring this requirement into focus. Computer networks are going to make a corporation's data bases more exposed and accessible not only within a corporation but outside a corporation as well. The simple fact is that companies must, for competitive reasons, become increasingly dependent on large volumes of information stored, manipulated and transmitted by computers. This requires that users demand better security on their data. The increasing incidence of computerized crime for personal gain as well as for vindictive destruction also points out this need for better security. One small part of the entire security picture is the technique of encryption of data. Similar in concept are the techniques of data compression which one would employ either for reduction of storage or transmission time. By using suitably programmed techniques in a general purpose computer, one can currently accomplish suitable data transformations, for purposes of either encryption or compression. The problem is the excessive additional cost and time this would add to data base processing particularly those data bases whose activity is moderate to high. Users will require more efficient techniques. The most apparent way that processing for data transformation will meet this need is the development of specialized hardware assists for software modules. Alternatively, data transformations could be all done in independent hardware encryption or compression devices. With the advent of specialized data base computers, this type of hardware would logically become part of that architecture. Other areas where specialized hardware may be employed are the transmission lines and I/O channels where information is in transit.

A user requirement which will only be met by the slow evolution of data base technology is that of data and program independence. Suitable methods of describing data, independent of program logic, have not been developed yet. Users are continually adding new data base applications or modifying existing ones. This normally means that structures or relationships of data should be modified. With most heavily used data base managers, the users must compromise between the best way to solve their new problems and the constraints of having to modify many existing programs. Programs <u>are</u> dependent on data structures. Frequently the user's course of action is to warp his problems into the existing structures because the investment needed to change old programs is simply too large.

Another aspect of data description is the user requirement to convert a data base from one form to another given that a description of the source data base and target data base exists in sufficient detail in machine readable form. To carry out the above processes requires both a logical description of each data base as well as descriptions of the logical to physical mappings. Research work is going on to figure out how to rigorously define these descriptions and mappings. Then translators must be written to interpret these descriptions and mappings to convert a data base from one form to another. This enumeration of

the problem is grossly oversimplified. Many of the current data bases are an amorphous combination of data description, both explicit and implicit, control languages, procedures, algorithms, actual data and program declarations. Thus, for grossly different data bases, the problem is extremely difficult. Users do carry out data conversions now. For each conversion from a particular source file to another particular target file, a special program is written. This program generally has complete knowledge of the structure of both the source and target files. No general purpose techniques are currently available. As computers become tied together through networks, the users will have increasing need to make these data base conversions. Different techniques will be required for those cases where conversions take place very frequently as compared to the very infrequent or once only conversions.

CONCLUSIONS

Data base technology is in its infancy. User requirements and the changing hardware architecture and circuitry will bring vast changes. User requirements will, in turn, be shaped by the availability of new hardware and software technology.

DISCUSSION

STEEL:

One thing that you commented on is the issue of transferability: moving from system to system, from hardware to hardware. Before having a sensible discussion on transferability we must define exactly what we mean. Maybe it's fair to say a system is transferable if you can move it from one chunk of hardware to another at no more than 5 per cent of the initial development cost. I would also suggest that, accepting my definition, it's going to be 1985 or later before we get there, although I admit it's a desirable goal. You were also talking about pictures and rotating things and giving different geometric views. Is that transformation part of the data base system or is part of the application program?

JOYCE:

In the example I was talking about, it's part of the application program. I'm not saying it should be, but that's where it is.

SHEEHAN:

The first question has to do with data independence. We can say that the vendor should providing us with the capability of reasonably restructuring data via DBMS capabilities without perturbing applications. That doesn't happen to be true now except in very, very limited restructuring areas. What do you see as the bounds that you would reasonably place on restructuring of data bases in your own shop in terms of both changing structures, adding elements, inserting elements within the structure rather than at the end which most systems can do?

JOYCE:

You must do an economic analysis of the cost of reprogramming versus the cost of additional overhead one would have to pay in a normal daily operation, trade those off and find the cross-over point. The only way you can look at these things is to bring them down to: how much money am I going to invest in this facility and how much am I getting out of it. You have to put a dollar value on computer time, programming costs, on what happens if you fail for an hour, what happens if you delay an application for some period of time because

you can't easily restructure.

GOSDEN:

I'm disappointed in very long shopping lists, but is there any way that you can pick out a short list of up to six things that would give a high pay off to you.

JOYCE:

One of the high priority items would be that of program and data independence. We really do spend much money and support many people in the corporation whose job is maintaining and converting applications because the data base structures are changing.

METAXIDES:

I'd like to respond to the question that Diane Sheehan asked. First of all you need two descriptions of the data, one describing the data base and one describing the data to a program: the Schema and Subschema concept. The next question is what mappings should be possible so that having written a program I can change the structure of the data base and have my program continue to work. I like to be able to add data items to a record, to change their length and their type, to combine data from several records, to be able to divide one record up into several other records, to add sets, to combine sets. Provided you're willing to pay the price and the continual drain of the mapping every time you execute a command, almost any mapping is possible. I'd also like to have the ability for the data administrator to specify his own procedure to provide additional mappings. If the basic architecture includes Schema/Subschema concept, then the actual mapping mechanisms can grow gradually.

BACHMAN:

To what extent is the problem of recompiling important? Are you really being that independent if you force recompilation of a program? Is that an acceptable answer or does true data independence mean it should run in its object form?

JOYCE:

If you could recompile and get data independence that would be a significant step forward. It would be acceptable for a few years, I would think.

MOEHRKE:

We're striving for a rough degree of object time binding. We believe that it is desirable to remove as much of the semantics as possible from the users application program. There are certain semantics that are rather natural to the application, there are some that may be peculiar to a particular DBMS. By building an appropriate level between the application programs and the particular DBMS, by identifying those semantics which are peculiar to the application and those which are peculiar to our particular DBMS, I feel that one can make a great deal of progress. We have successfully plugged in and unplugged IMS under our interface with no apparent problem. It can be done with today's interface.

MELTZER:

The suggestion was made that data independence is achievable with one level of indirection, the Schema and the Subschema. I suggest that the Guide-Share report indicates quite properly the two levels of indirection necessary.

STOCKHAUSEN:

I agree that there seems to be a lot of sematic information embedded in the structure in the data base. Producing data independence by the Schema and Subschema structure takes care of the layout within a single record of information. There's still a problem when you interrelate between a parent and a descendent, if you build application programs which are sensitive to the structure between the parent and the child and which know they have to go down exactly one step to get the next logical unit. That prevents changing the data base structure. If we want to bring the structure out into the open, we must allow the data base management system to decide how many levels are required to get from here to there. The application programs that result are just simple processors that output data or input data but perform no procedural action on the data.

Data Base Management Systems, D. A. Jardine (ed.).

USER REQUIREMENTS FOR DATA BASE MANAGEMENT SYSTEMS (DBMS)

H. S. MAYNARD
EXXON International Company
New York, New York

INTRODUCTION

This paper will develop and present user requirements in the area of Data Base Management Systems (DBMS). These requirements will be based on Exxon International's experience to date, as well as what we see in the future. The intent is that these requirements will be broad in nature, and reflect the needs of our business. We do not have preconceived solutions to satisfy these requirements in mind. We are, of course, addressing these requirements ourselves; but we are short, far short, of the resources required to satisfy them completely by ourselves. We believe that these requirements are not unique to us and, therefore, there is a market for the solutions.

This paper will begin with some background information, and then present a brief description of some of our present systems. We will then discuss the future systems we see coming along and the implication of these systems, present and future. With this buildup, the requirements that we see at this time for DBMS will be presented. The conclusion will briefly outline some of our efforts we are undertaking in addressing these requirements.

BACKGROUND

Business - Exxon International's Role

Exxon International is a special-purpose organization which forms more or less the hub of Exxon's near-term international operations. We have operating, co-ordinating, and advisory activities, which broadly include serving as the supply and transportation link between Exxon's regional companies, international marketing and certain centralized services. We are a commercially oriented organization actively trading in tanker and oil markets.

Our principal logistics activities consist of developing and implementing the Exxon worldwide supply operating program for roughly the two-year-ahead period. We also negotiate arrangements for inter-regional crude oil and product purchases, sales, and exchanges; and develop recommendations as to inter-regional prices for crude oil and products.

Exxon International's principal international-marketing activities consist of developing, negotiating, and implementing bulk cargo sales of crude oil to non-affiliated customers who are in a position to shop worldwide markets. We also provide guidance to regional organizations worldwide for the marketing of aviation and marine fuels and lubricants.

Our principal centralized service activities consist of servicing two fleets of affiliate-owned tankers, and of chartering tankers in the market place for the worldwide tanker needs of Exxon's affiliates. We also design, contract for construction, and supervise construction of oceangoing ships for Exxon's affiliates. We also provide certain centralized accounting and financing services.

Exxon International's main thrust is to serve the interests of the Exxon family. We are very much concerned with current industry commercial developments and current operating needs and problems. However, we must always be sure of both the long and the short-range impact of our actions and recommendations on regional and other operating organizations. We depend on meaningful communications, interactions, and mutual understanding with these other regional and operating organizations.

Technical

One of our main operating principles is to try to keep our technical environment simple. By this, we try to keep to a minimum the number of languages, the type of terminals, the number of vendors, etc. This enables us to concentrate on applications. The vast majority of our systems are a combination of DL-1, COBOL and MARK IV. The on-line aspects of our systems use IBM's IMS system. We have been in production on-line under IMS for some three years, and are currently operating under the latest release of IMS.

Our computer resources consist of an IBM 370-155 and an IBM 370-165. The IMS system occupies the 1.5 million byte 155. The 3 million byte 165 is the work horse of the computer centre and runs everything else. We have two 3780's RJE stations in New York linked to the 165 at the Data Centre in New Jersey that we use for development work.

Our IMS terminals, about 110 are almost all IBM 2260's. We have tried and are using a few plug-to-plug 2260 compatible CRT's with mixed results. Also our experience with IBM 3270's to date is mixed. All in all, we spend more time with terminals than we think we should have to. They are a major pain.

Systems Function

The role of our group is that of systems applications. This includes the development of systems requirements, evaluation of alternatives, development, implementation and maintenance. This is a fairly classic role for an application function. We have no responsibilities for any part of the computer operation. Our Data Centre is 60 miles away in New Jersey and is shared with other users. We are presently responsible for about 40% of the work processed at the centre.

Our organization has a total of 37 people. This includes 3 managers, 2 secretaries, and 32 professionals performing in the role of Project Leaders, Systems Analysts and Programmers. Our staff has been fairly constant for the past two years and we expect to keep it at about this same level in the future.

TYPE OF SYSTEMS

Present

When describing the type of systems in use in Exxon International, we often use the phrase "operating information system". This phrase describes information systems that contain the basic information of a certain business function and are an integral part of the operation. This type of information tends to be very detailed. For instance, we have more than 100 data elements of information describing a single tanker. This information is used in scheduling and dispatching operations.

Another characteristic that will help describe our systems is that they are used mostly by the people in the lower-level jobs of the business function. In our organization, however, these are mostly professionals. In some of our systems, there are some first-level management people who interact directly with the system. We do not, however, refer to any of our systems as "management information systems". We are just now beginning to implement operating information systems in the accounting area that are mainly used by clerical personnel. At this time, accounting personnel represent only 10-20% of the users.

The following section will describe several of our "operating information systems" and what operations they support. This discussion should be helpful in developing an appreciation of the role of systems in our company.

Logistics Information & Communication System (LOGICS)

LOGICS is the system which is integrally involved with Exxon International's responsibility for coordinating the inter-regional movement of crude oil and refined products. It is used as a "tool" to help the decision making, communication and clerical tasks in the operation. The functional areas that now use LOGICS include: planning and scheduling of oil maovements (slating); allocation of vessels; dispatching tankers and tracking their itineraries: chartering of tankers; operational monitoring of slate status,

oil contract performance and vessel performance.
LOGICS has the capability for supporting simultaneous operational links to Exxon regions and to major suppliers and customers involved in the supply function. Exxon USA (Houston), Esso Europe (London), and Esso Eastern (Houston) supply functions are currently linked to LOGICS. There will be other links in the future.

Aviation Data Bank System (ADBS)

The Aviation Data Bank System was created to provide marketing information for use in assessing market strategy and Exxon's position at airports and in the development of Exxon's business on both long and short term bases. The information obtained from the system is based on data on: airports, airlines, and customers; Exxon bids and those of competitors; prices, contracts; results of short range planning and analysis, marketing sales and plan of action.
ADBS interfaces with the Financial Control System by providing FCS with the ability to verify accounting entries against the ADBS Airport information.

Cargo Trading Information System (CTIS)

The Cargo Trading Information Systems (CTIS) is a marketing system for the bulk cargo business that incorporates the development of short-range forecasts and communication activities between the Cargo Trading and Supply and Transportation functions. CTIS has been designed to meet specific performance monitoring, forecasting and reporting needs of the Cargo Trading Operation by integrating outside purchase/sales contracts, reference plans, monthly operating plans and performance data into a formatted structure.
CTIS interacts with LOGICS as previously described by providing contract planning and performance data. It also feeds price information to the Financial Control System for the invoicing operation.

Financial Control System (FCS)

The Financial Control System is a cash and receivables management system whose objectives of control and planning are attained by rapid and accurate data gathering and entry facilities. There are also capabilities for rapid retrieval and display of organized information concerning cash, customers and affiliates accounts. A centrally-located computer data bank of current and historical data facilitates reporting and forecasting.
The direct entry capabilities of FCS have been enhanced by its ability to verify accounting entries against information in the Aviation, Cargo Trading, and Marine Sales systems.
Additional interaction with LOGICS includes the feeding of actual loading data (volumes and dates) into both FCS and LOGICS data bases. This actual data is received via cabled messages from worldwide ports.
These brief descriptions give an overview of some of our major systems. There are many major business functions of our company that rely greatly on the computer. As significant as these systems are, however, we feel that we have a long way to go before we will satisfy all of our needs.

Future

Much of our future efforts will build on our present systems. In addition, we will be implementing systems in areas that today make no use of the computer. There are two aspects of the future that we would like to briefly discuss. We are sure there will be a great deal more than this, but these will hold us for some years.

Multi-function Information Sharing

As described in the previous section, the ability to share information between various users in a major objective of our systems. While we have made substantial progress in the area of information sharing, most of that information sharing has been between different users within the same business function. The users of the Aviation system are all Aviation marketing people; the LOGICS system

is used basically by Logistics people, etc. We are just now beginning to develop cross functional or multi-functional information sharing. The major area for this is the tieing together of the financial systems with both the logistics and marketing systems. Reference to this sharing is made in the specific system descriptions. Previously the financial operation usually had copies of such items as customer contracts, prices, volumes delivered, barrels to tons conversion factors, etc.

Needless to say, these copies did not always agree with the information in the operating function. There have been endless efforts to reconcile items like sales recorded by the financial people with the figures used by the operating people. The use of different barrels-to-tons conversion factors, for example, can cause frantic hair pulling. It may be difficult to understand that we have had these difficulties so long. An example may help: the billing of a tanker-load of oil occurs as soon as possible after loading. The information for billing a cargo out of the Persian Gulf goes through several companies, countries, communication systems and internal Exxon groups. The type of information involved with a loading in tons, sulfur content, specific gravity, residual cargo on board, dead freight, time and date of loading and customer to be billed, etc. When this loading information is received by the invoicing function and added to certain contract information, the invoice is prepared and mailed.

To continue the example, the tanker may take weeks to reach its destination. It is not unusual for the destination of a tanker to be changed to respond to changes in inventories, demands, etc. When you think of a voyage of weeks from the Persian Gulf to Europe, you can see the need to wait as long as possible to decide where (customer) in Europe to deliver it. These changes are made under the extreme pressures of the scheduling operation. Sometimes the notification of a change does not get passed on to the invoicing people.

Sooner or later, of course, all of these changes are picked up by the financial people. But between the time changes are made and that "sooner or later", there can be a great deal of reconciliation. The answer to this particular problem is currently being implemented as a part of Financial Control System. In preparing an invoice for a transaction as described above, the type of operating information needed for invoicing will be "pulled" from the operating information system. The appropriate financial information added and an invoice produced. At the time the subset of operating information is "pulled", the data base will be updated as to the fact an invoice has been produced. This will allow the system to "notify" the financial people if any change that affects that invoice is made. This, incidentally, is an example of our "change notification" system which will be discussed later in the section on Data Base Requirements.

Another major area for multi-functional information sharing will be in the areas of monitoring. We use this term to describe such business tasks as "monitoring" plans versus actuals, and contracts versus plans versus actuals. Our various lines of business have rather symmetrical planning and operations organizations. Today most of our information systems are on the operations side of the business. As we develop systems in the planning areas (see next section), this mutual sharing between planning and operations will become even more significant.

When you add up the sharing of information between the planning and operating functions of a line operation and the planning (financial forecasting) and operation (invoicing, etc.) of the financial organization, you really give the information a work-out. When you then add another dimension of sharing between headquarters and some organizational subset (region, district, etc.), you begin to see why we must solve our problems of control over shared information systems.

Planning Information Systems

As mentioned several times before, most of our systems today are in the operational areas of our business. We have, more or less, the same number of people in the planning areas as in operations. With the exception of several significant models (they can be counted on two hands), this planning function is carried on without the aid of computers. It is hard to imagine that the planning operation cannot be assisted by computer systems. The benefits of improved

planning, in any company, are tremendous. They usually are greater than the benefits of a similar degree of improvement in the operational area. Why then have we not made inroads in this area? We in Exxon international do not really know the answer to this question. Our current thoughts are that we will need vastly-improved techniques of interactive modelling capabilities tied to elaborate data base systems. Also terminals with increased alphanumeric and graphic capabilities and with significantly increased computer power readily available to the end user. If we look at current computer capacity being used today for inquiries, updates, simple reporting and compare that to what would be required to handle the same volume of transactions all asking "what if" questions, this future need is apparent. The key will be effective computer power.

We are starting to address this question specifically and to develop an understanding of what is needed in the planning areas of our business. The results of this effort will not be known for some time. It is impossible to know what impact our findings will have on DBMS requirements, but we feel they will be significant.

Implications

When our present and future systems are looked at as a group, there are several aspects that warrant further discussion. These items are all related and will help develop the background upon which we will present our DBMS requirements.

Integral Systems

As noted in the discussion of specific systems, many of our systems have become an integral part of the business function. This has, of course, required a great effort by functional people to define the requirements for these systems. It also, then, requires a great deal of their time to keep the system viable as an integral part of a changing business invironment.

We have found that if a system is going to be an integral part of a business function, it invariably has some on-line components. In our case, the majority of our IMS data bases are on-line. We have developed an environment of "on-line systems". By this, we mean that even "small" systems are now being put on-line. In many cases terminals exist and development and operating costs do not vary significantly from batch. It is no longer a "big deal" to be on-line.

Reliance on Computers

The consequence of these systems becoming an integral part of our business functions is the heavy reliance on the hardware and software. The concern is expressed by constant questions, such as "what happens if..." and "has such and such ever happened?" We find ourselves involved with studies on standby power, catastrophic backup and alternate communication facilities. We become more conscious of physical security, passwords, etc. These concerns will always be with us and will require constant effort. In many cases, we cannot quantify the impact of a "system down" situation but the one thing that is clear is that whatever the "cost" of a failure is, it is constantly increasing.

Information Sharing

Information sharing is not an entirely new aspect of computer systems. We feel, however, that we have added some new dimensions. Every reader is familiar with the typical ststem that contains a file update and the production of multiple reports that are used by various people and groups. The information used to create these reports is certainly "shared". There are two conditions that usually exist in this type of system. One, the system is usually scheduled, at least the update, and two, the interactions between the various parts of the system are controlled by people. By this, we mean the scheduling of the programs, the checking of edit messages, the handling of rejects, the checking for "end of job" the control of totals, etc. The type of "information sharing" that we are now involved with is where a common data base is viewed and updated from different terminals on a random basis. This sharing can be done by various parts of an organization and even by people or groups that are geographically separated. This is done without human control (of the actual usage, at the time of use) or

schedules. The implications of this type of sharing in the area of Data Base controls will be discussed in the section on DBMS Requirements.

The benefits of information sharing in this type of environment are worth developing. The major benefit is that of user confidence. The confidence of the users in an information system where he knows that the other users are working with the same, exactly the same, information is tremendous. For example, in developing the economics of placing terminals in London for our Aviation marketing system we were forced to place relative values on "degrees of sameness". If we had terminals in New York and London, the information would be "100% the same" and understood by the users to be identical. The cost was, of course, high. The alternative was to transmit changes or the complete status of the information each night. This would have resulted in a "very high degree of sameness" but something less than 100%. The cost of this approach would have been substantially less. It was felt that if there was a chance that one of the users had different information (and maybe "better") from the other, the system would breakdown. There would be phone calls to check, separate records to keep, etc. As you will recall, this system is used by professionals and management people in the marketing area of our company where they are always looking for one more piece of information before submitting bids, establishing price, etc.

The important point is the understanding and confidence of the users that the information he is viewing is identical to that seen by the other users. The only way we know to share identical information is with remote terminals connected to a common data base.

Another benefit of shared information systems is that both the accuracy and the precision of the information is "improved". This comes about because different users all have slightly different needs, perspective and areas of importance. This is then reflected in the degree of accuracy and precision required by each user. Reconciling these differences results in the degree of accuracy and precision required by the most demanding user being available to the other users. In most cases the other users may not need this "improved" quality of information, but it is usually worth something.

The same kind of argument can be made for the timeliness of the information. Each element must be as current as the most demanding user. In satisfying his need, it is of course available to all users in the same time frame. More current information always has some incremental value.

There are various related costs that are associated with the benefits of shared information systems. An old problem that becomes even more important is to know what information is in the data base. The reconciling of definitions is common to all of us. In our business, the "same" product (as far as manufacturing is concerned) is put into two different packages and becomes "unique" products from a marketing viewpoint. We even have problems with geography. In the country table for our Marine Sales activity, we have something called "Benelux". This does not exist for other segments of our business. Knowing what information is in the data base is not only a problem of definitions but also of status.

The question of the status of the information in the data base drings up the problem of "change notification". This refers to the need for the system to initiate some actions and actually "notify" the appropriate users of the system of recent changes that he should (must) know about. This is a major "cost" of shared information systems and will be more fully described in the section on DBMS Requirements.

A more mundane problem is that of developing an equitable system for allocating the costs of the system to the various users. This is a new aspect of an old problem.

On balance, we feel the most important aspect of the various systems we have installed to date is "information sharing". As we have defined it, this requires on-line terminal-oriented systems. We started with on-line systems to be able to keep certain data bases up to date. It is important to know the current position of our tankers, our orders, inventories, etc. We now see that these systems have the capability to provide "information sharing" and this has led us to such uses as described above in our Aviation marketing function. The marginal economics of

adding another user to an existing system can result in very impressive returns.

DBMS REQUIREMENTS

The previous sections of this paper have discussed our systems, present and future, and their implications. While some of our business functions may be unique, we do not feel that our problems are. We feel that most of the problems discussed are recognized by the reader, either as present concerns or ones he can see in the future.

We have combined our needs into three areas. The first, system monitoring, is brief and relatively well defined. The second, data base control, is less well defined and is the major part of this section. The third, increased efficiency, is a general discussion.

System Monitoring

The "system" we need to monitor is the entire system which supports our production information systems. Included in the "system" are the main computer, software, phone lines, modems, terminals, control units, etc. The factors needing to be monitored are cost, availability and response time. These factors are, of course, related. Cost is usually, but not always, the offset to the other two. The cost of the system is the easiest to monitor and is usually available and accurate.

Availability is a more difficult problem. We need to monitor the degree of "availability" to the end user, at his terminal. There are also several dimensions of this monitoring: by terminal, by application and by location. The availability should be described in a normal statistical manner. We are making slow progress in this area, but usually only after a user complains about his availability. It sounds like a simple straightforward effort would satisfy this need but we seem to be always lacking.

Response time is probably the most difficult factor to monitor. We have various estimates such as CPU residency time, but they are merely that -- estimates. We recently had six people sit with stop watches and sample response time with the resulting accuracy you would expect. We find that by not really having a valid measure of response time, we are at the mercy of the user who says "my response time has lengthened". This user perception may or may not be true, but such a situation is a negative factor on the creditability of the entire system.

The ability to monitor the system properly should be relatively stralght-forward. Our inability to do it properly results in a poor image. In addition, we find ourselves on the defense and disagreeing with the various people involved in the system over what is fact instead of working on improvements.

Data Base Control

In this section, we will discuss five data base control functions we feel should be provided to the application developer. In our case, these are the kinds of features that IBM's IMS system should provide because they are basic (universal) functions for our current of future applications.

Today these control functions are the responsibility of each application. Our belief is that they should be separated from the individual application problems. The need for these control functions are not unique to terminal-oriented information systems; however, we feel that the need for these functions is greater where the interaction with the data bases is not controlled by people. Rather, as described earlier, the interactions are of a random nature.

Security

Security relates to identifying who is using the system. The who can refer to a person, terminal or group of terminals. We will not develop this area as the needs are well recognized. It is included for completeness and because it relates to some of the other data base control functions.

Authorization

Authorization is an extension of the security function. After the "who" is determined, the next question is one of authorization to do "what" with the Data Base. Is the user authorized to change the data or just view it? The answer to that question can be a function of a value in a field, a range, a combination, a function of time, etc. The answer can also vary depending on whether the change calls for an addition, a modification or a deletion.

Authorization is not a new control function for computer systems. Again, the sharing of information between multiple users causes us to get more specific in many cases. We find that the logic to answer some of these authorization questions requires references to individual data fields and values recorded in them. In some of our older systems, the question of authorization was usually a function of a file or job number.

Validation

Validation refers to the checking of the change to determine its validity. It is a familiar problem that knows almost no limit of complexity.

The above three control functions: security, authorization and validation, have one need in common - the handling of exceptions or errors. All three can really be considered various forms of "validation" and require various methods of checking against tables, logical routines, etc. The notification of exception to these validation steps can take the form of a message to the originating terminal, a message to the audit log, a message to another terminal, etc.

All of the above data base control functions precede the actual change or use of information in a data base. After a transaction passes these control functions, the following functions are invoked:

Logging

Logging refers to documentary logging. (This is usually separate from the logging that is used for systems recovery.) The most typical type of documentary logging is to record the before-and-after status. The significance of the change is then available to other users. Another kind of logging is that which occurs at a certain time such as day-end or month-end to record the status as of that time.

In addition to portions of the data base information, a logged transaction may include summary information, time and data of the change, identification of the person making the change and an area for comments. We have found that the allowance for free-form comments can be very important in systems of this type. (It is good to be finished with the 80 column card½)

There is another aspect of logging that presents itself, particularly when data bases are shared. An unshared data base can be changed based on its own rules and needs. When it is shared, the rules and needs of the other users must be considered.

We find that in many cases, the person making the change is really "requesting" that a change be made. This request must be approved by other users or organizational units before it can be posted to the shared (agreed upon) data base as a bonafide company transaction. In this case, a pending status situation must be included in the data base. Thus, our view of a data base shared by many users includes three subsets of data: currently unchangd, changed and agreed upon by all, changed and pending agreement by one or more users.

Access to the shared data now has another dimension or set of rules for each user. How do I want to access (view, total, process) records in the data base based on their "change" status?

Change Notification

The need for change notification results primarily from sharing information between multiple users. When a change is made to an unshared data base, the transaction is completed after validation. However, when a change is made in a shared data base, the system takes on a new responsibility: to notify the other appropriate users of the change.

The "change notification" function is usually initiated by or triggered by a change or proposed change to a data base. The notification action can further be qualified by such things as who made the change, the degree of change, specific values, or the time the change is made. The function can also be initiated by the fact that no change was made under certain conditions.

Once the system determines that a "notification" is required, it must then be scheduled. There are several options available. The notification can be immediate or delayed. The delayed option itself has several alternatives. These include notification at a pre-scheduled time, after a certain elapsed time, upon command of either the person making the change or the person to be notified. For instance, a person can request to see "all the changes made since 10:00 this morning" that affect him.

The type of notification can take many forms, messages directly to terminals, an update to the pending segment of the data base, a batch report reflecting new summary information, etc. The response to this notification itself falls into two basic categories, one where approval of the change is required and the other when the notification is for information only.

The above brief outline is not meant to be all-inclusive of the various aspects of the change notification function but merely to indicate its complexity. We do not completely understand all the implications and requirements ourselves. In fact, it was not very long ago that we first started addressing it as a specific requirement in the control of shared data base information systems. One way to look at these data base control functions is as a system by itself. This is really an information processing control system analogous to a physical-process-control system. It is controlling the processing of information in a continuous operation. The analogy becomes clearer if you think of the system running 24 hours per day with terminals accessing the information on a random basis from many remote locations. We have found this idea - that of an information processing control system - useful. An example of this will be covered in a later section covering our efforts to monitor the system. (A chart depicting this concept is in the appendix.)

Dictionary

As time goes on, more and more is written about the illusive term, "dictionary". As we use the term, we mean to describe the tool that will be used to control the data bases. One line of reasoning even indicates that the dictionary itself is (will be) the "central corporate data base" and not the data bases themselves. In any case, it is generally agreed that the dictionary will contain such items as data and code definitions, security features, use statistics, standard validation criteria, etc. We do not plan to elaborate on these features, but just add a few to the list.

When we think back to the previously described "information processing control system", we need to be concerned about the "rules" that guide such a system. It is our position that these rules belong in the dictionary. Further these rules must be available to the various users. For instance, if a user wants to know the authorization rules for a certain type of change, he should be able to have them displayed. The form that they should be displayed in probably will be tailored to the user and not just a coded decision table type format. They should be understandable to a non-technical person and available directly to him without a middleman. This is imperative if the users are going to have confidence in the types of information systems we have been discussion. They must know that such and such will happen if a certain change is proposed. These rules are really the rules of the business, the way the details of the business are carried out, the way the information flows within the business.

Another facility relates to these rules: they must be adaptable to reflect the changes that occur daily in any business. Two more points follow from this facility. First, the dictionary (particularly this area on rules), must have its own control system; second, when a change is made to the rules, we must be able to analyze the status of the information in the system, pending changes, etc. to determine the impact of the change in rules to the current status of the Data Base (e.g. do the new edit rules make current data invalid?)

These last few points are far from fully developed or even understood. We feel it is important, however, at least to raise them when we are talking about our requirements for data base systems.

Increased Efficiency

Here is a general plea for techniques and tools to increase the efficiency of the system development and maintenance function. We see demands grow as the time to implement decreases. When we see how much effort is required today just to keep our systems current to the changing business world, we have concerns for tomorrow. The need to keep current is becoming critical not just to keep the systems operating but to keep the business operating.

We are looking for a real break-through in this area. A systems group like ours is really nothing more than a "translator" and as such represents an element of inefficiency. The ideal case would be if the users could interact directly with the computer system and not rely on a middleman, us. The end users and the computers should really "speak the same language".

WHAT ARE WE DOING

It might be useful to finish this paper with a sample of some of the things we at Exxon International are doing to help ourselves. The discussion of our efforts is organized in the same manner as the previous section.

Systems Monitoring

We are actively developing a plan to install an IBM System 7 Sensor Base control computer to monitor "the system". It will initially monitor availability, response time and usage. We will, at long last, have credible information on system performance. The information will also be useful in two types of simulations: one will help us optimize the number of terminals per control unit, control units per line, etc., and the other will allow us to simulate future changes of many types in the use of the system. Today, we are not very sophisticated in our approach. We have a simulation model of the system but our data is not very good. It is also difficult even to validate the model. One of the specifications for the System 7 is to provide a data base of system monitoring data directly into a simulation model.

Data Base Control

There are two items that fall into this category. They are briefly outlined below:

Change Notification System

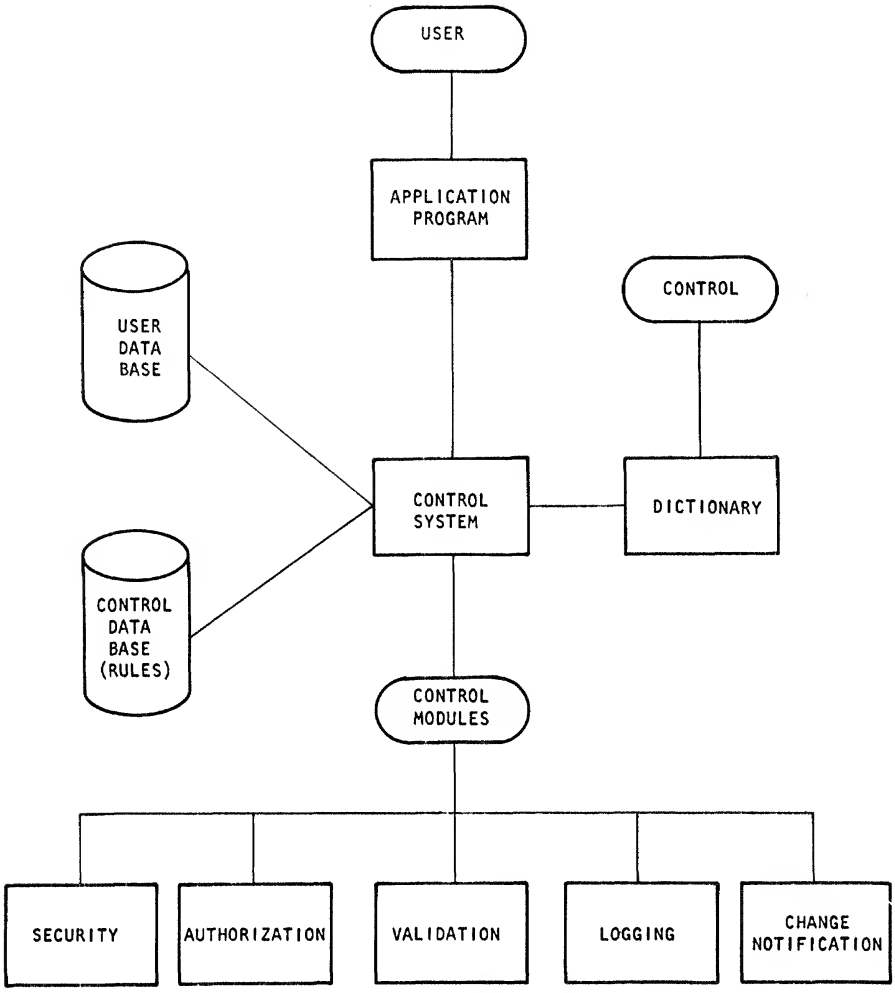
We have in operation a system which keeps track of changes to a shared data base. These changes are kept on a separate data base in a "before and after" status. They are available for the same type of inquiries and searches as the base information. We have been implementing this step by step; we are learning as we go.

Dictionary

There are several aspects to this effort. One is a survey of available systems that has been started to lead us to a "make or buy" decision. We have also started to develop an inventory of such items as screens and tables. We are beginning to implement a system to map the use of each data element in our data bases to the specific screens that use them. This mapping will be an integral part of the system and not rely on external documentation. We feel this is the area to begin with and will result in a very significant step towards the control of our data bases.

Increased Efficiency

There are several items of interest that are aimed at increasing the efficiency of our systems development efforts.



CONCEPT OF AN INFORMATION PROCESSING CONTROL SYSTEM

Acknowledgement of the Problem

This is the most important step. By having the problem of efficiency paramount in our minds, all day-by-day decisions, as well as the planning efforts, can be measured as to their impact on overall efficiency.

User Training

We have been encouraging the system users to do more and more of the tasks that we have done in the past. These efforts include report-writing, dealing directly with the data center on production problems, developing system requirements, etc.

Organizational Changes

We have made an attempt to keep our organization more flexible than in the past. This flexibility relates to the overall organization as well as the makeup of individual project teams. In the past our group was organized more or less in parallel with the company. We now find the growth of information sharing causing us, in some cases, to organize across several Company departments.

General Purpose Software

Two years ago we initiated an effort to determine the common components of our various systems and develop "a better way" to handle them. We are now testing a system, called IPG, which is a "better way". IPG stands for Interactive Program Generator and is briefly described in the appendix of this paper. The system is significant because it addresses both the requirements for increased efficiency and data base control.

132 Character CRT Terminal

We include this item as an example of the breadth of our efforts to address efficiency. When the CRT terminals we are now using are looked at in terms of efficiency, we ask such questions as: why should we have to design and program differently depending on the output device? The 132 character, 60 line printer seems to be a standard, why not the same for a visual terminal? Our answers on such questions, to date, are not very encouraging. Most vendors seem to think we are mixed up.

SUMMARY

We have presented the major problems and concerns that Exxon International faces today. They are based on our experience in developing and relying on information systems that have become integral parts of our various business operations. As such, they are subject to the same sorces as the business is. These include such things as security, concern for efficiency and the need to respond to rapid changes. Becuase our systems are all data base systems, we see the solution to these problems and concerns as "user requirements". We realize they are broad in scope. While we are addressing these needs ourselves, in a limited manner, we do not have predetermined solutions in mind. We would expect the solutions to come in the form of both hardware and software, at least as we see them today. Some of our business operations may tend to be unique but we do not feel that our requirements are unique. The business problems we face are common to all.

APPENDIX

INTERACTIVE PROGRAM GENERATOR IPG

Background

In spite of the advances made in the capabilities and facilities of higher-level languages such as PL/I, COBOL, etc. and of data base management systems such as IMS and its DL/I facilities, the development of an IMS application in a

CRT environment still involves a significant amount of repetitive and detailed coding. It is not uncommon to find that a trivial program which does little more than receive and validate an input screen, issue some data base calls and return an output to the CRT terminal requires several hundred lines of code. A random survey of 10 on-line programs in an order processing system developed by Exxon Chemical Co. revealed an average program size of approximately 650 COBOL statements, of which 370 are declarative DATA DIVISION statements and 270 are procedure PROCEDURE DIVISION statements. Moreover:

- A non-trivial amount of the procedural statements deal with DL/I calls, the building of Segment Search Arguments (SSA), post-I/O status checking and input validation and error handling.
- Most of the data declaration statements involve the layout of input/output areas for screens, data base segments and control blocks required by IMS for the Program Communication Blocks (PCB), Segment Search Arguments (SSA), DL/I function codes, etc.

It was apparent that much of this trivial but detailed and error-prone coding could be generated from some key specifications from the analyst.

Therefore, IPG was conceived as such a programming tool. Its objectives are basically to:

- reduce the coding details heretofore necessary in the development of every IMS application.
- increase programmer-analyst productivity.
- reduce the IMS application development cycle and cost.

General Description

IPG is a generalized CRT-oriented programming tool designed to aid the analyst in the development of CRT-oriented IMS applications using DLI data bases. The major features of IPG are briefly described in the following paragraphs.

IPG provides an application-oriented command language including:

- powerful DL/I-oriented input/output commands
- data validation commands
- data location commands
- data manipulation commands
- data logic control commands

A PL/I source program is generated by IPG from the commands and data specifications entered by the analyst. It is this PL/I source program that must be compiled and executed as an IMS application to achieve a working application. For user convenience, IPG commands can be entered via CRT not only in the conventional "coding pad" format but also in preformatted prompt displays.

Given the CRT display-oriented environment in which IPG and its generated programs operate, IPG provides the following display handling facilities:

- At application program specification time, IPG allows the analyst to define a screen format and specify the contents of the screen.
- At application program execution time, IPG provides a run-time Screen Handler module designed primarily:
  - to perform the mapping function between the external screen format and the internal program format for this screen data.
  - to provide numeric data checking/edit functions for use with those terminals (such as IBM's 2260 display stations) which are not equipped with such capabilities in the hardware.
  - to provide, in general, terminal-type dependent functions to relieve the user program from such considerations.

Although IPG is being implemented initially as an on-line alphanumeric CRT based programming tool, its basic design for Data Base or terminal I/O functions are general purpose. We intend to expand its use to other types of terminals (printer and graphics) and to generation of batch IMS programs.

In order to minimize redundant data definitions across applications and indeed within the installation, several data bases are maintained by IPG to provide storage and access to the specifications. Among these are the:

- Screen Data Base, in which the layout and definitions of the screen fields are stored by name. Contents of this data base can

subsequently be retrieved for purposes of (a) verification/ modification by the screen designer (b) viewing and usage (to enter/display data) by any user in the system. This data base includes the data used to generate the PL/I statements which declare the SCREEN storage area.

- Data Base Specifications (DBS) Data Base, in which the structure of each user data base and a complete description of each segment are stored. This data base needs to be established only once, using the IMS DBD deck enhanced by the addition of data required by IPG. Thereafter, each program which references this data base will automatically be provided with the appropriate PL/I DECLARE statements for each segment referenced in a Data Base Operation.
- Program Specifications (PS) Data Base, in which the traditional IMS PSB information is stored, together with the procedural specifications of the user program. All information relative to a program is either contained in this data base or is contained in other IPG data bases and referenced by this data base. Hence, after the analyst has interactively entered his application needs, this data base is used as the primary source of information required to produce (i.e., generate) the final program.
- Field Data Base, in which attributes such as data type, length, picture and valid value(s) as well as relationship with other fields are stored for frequently used fields. When such fields are dealt with in an IMS application, the definitions can be defaulted to the assumptions found in the Field Data Base.
- Table Data Base, in which validation/conversion tables are built and stored when specified by the analyst.

In the design and layout of the five IPG data bases mentioned above, allowances are made to include data which is not necessary to the operations of IPG but is envisioned as useful or desirable information in the event of the implementation of a data dictionary system within the same installation.

Overview

Figure 1 shows an overview of the use of IPG to generate an IMS application. Six basic steps are followed to generate a working IMS application.

1. Load parameters into IPG data base for the applications data base and program from IMS DBD and PSB decks.
2. Enter IPG specs via on-line displays for the application. Describe screens, fields, procedures, tables, editing, DB processing, etc. Where needed include in-line PL/I statements.
3. Generate PL/I shource program from stored specs on IPG's data base. Obtain annotated listing showing IPG macro statements and generated PL/I code for these macros.
4. Compile and link edit IMS application program.
5. Test the created load module for the application program.
6. Determine changes to application program and proceed to repeat steps 1-5.

Before any IPG commands can be entered, the Program Specification Utility must be used to define the program to IPG (step 1). As indicated in figure 1, the input to this batch program includes the PSBGEN deck for the program, together with additional data required by the utility. The same input can be submitted for PSBGEN, as the PSBGEN utility will treat the IPG information as comments. Program Specification data is stored in the IPG's Program Specification (PS) data base.

For each data base PCB to be referenced by the program, the Data Base Specification Utility must be used to enter data into the Data Base Specification (DBS) data base. This process, also, must be completed before the entry of IPG commands. As shown in figure 1, thy input to the Data Base Specification Utility consists of DBDGEN input which has been supplemented with data required by IPG. The IPG data will be regarded as comments when the same input is submitted for DBDGEN. The IPG statements define each DB field to be later

references in an IPG macro command.

The analyst is now ready to use IPG's on-line facilities to specify the logic and I/O processing of the application program (step 2).

The analyst begins by specifying the format and data field attributes of screens which the application user is to employ in using the application program. Data entered for the application's screens are stored in IPG's screen data base.

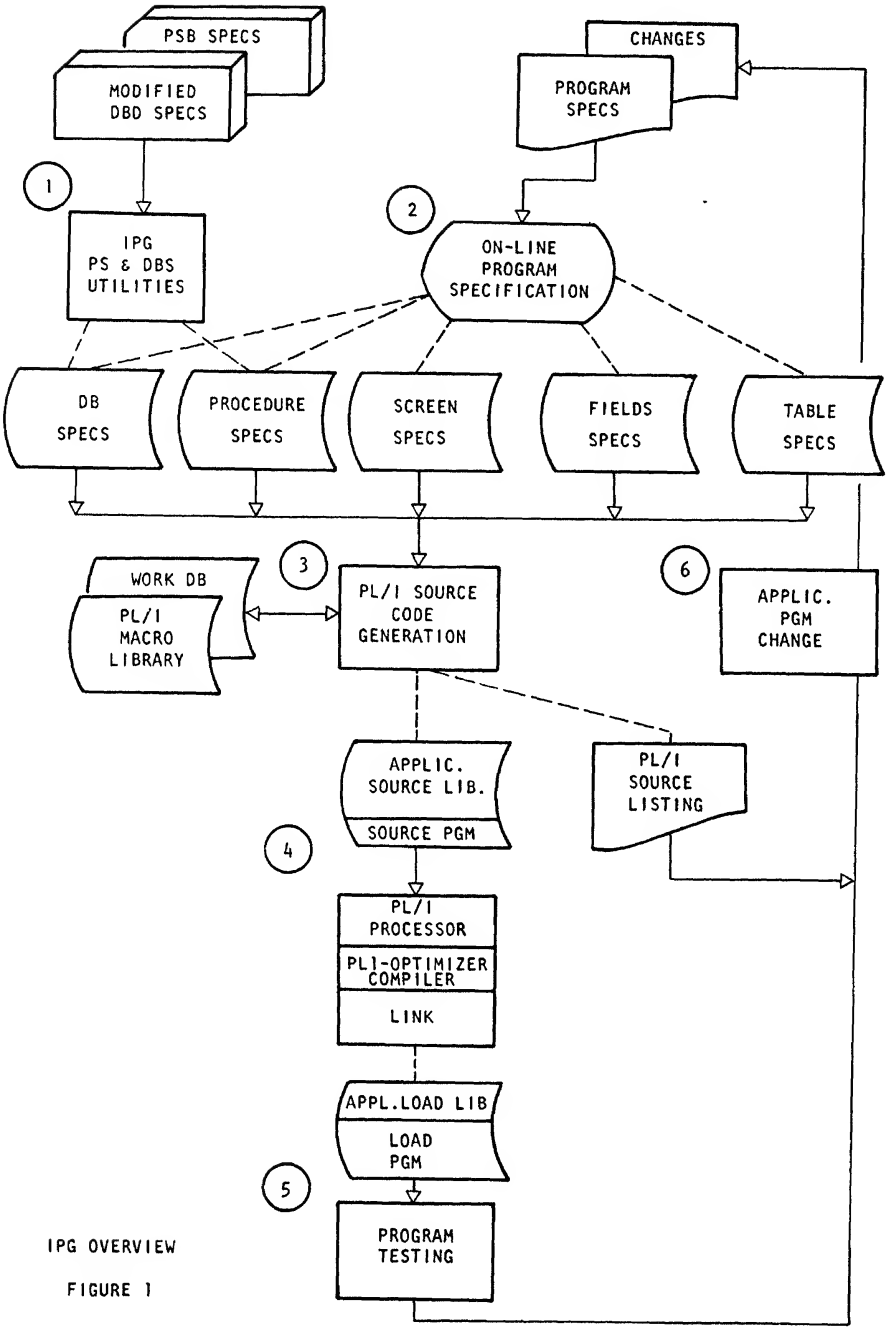
IPG names for data base segments and their fields can be established by including the defining information in the input deck for the Data Base Specification Utility, or by entering the data on-line after the corresponding DBE has been established in DBD data base.

To specify tables and enter coded entries which are to be referenced by and included in an application program, the analyst uses another IPG transaction. The table field definitions and data records are stored in the IPG Table data base.

Similarly, an IPG transaction can be used to define WORKING storage fields which will be referenced in a program. These field definitions are stored in the IPG Field data base.

The program logic and data base processing for the application are specified using a series of macro statements. More than 20 procedural macro commands have been defined for IPG thus far. These are entered on individual IPG specification displays and stored on IPG's procedure data base. Figure 2 gives a list of these commands and a brief description of their function.

After all data elements and program statements have been defined and entered into the IPG data bases, a batch program is used to generate an application cource program in PL/I source language as well as an annotated source listing (step 3). This output from the code-generator may then be submitted to the PL/I compiler (step 4). The development cycle is completed by testing and perhaps determining changes to the application program (step 5 and 6). Corrections and changes may be made to the current IPG specs for the application program by repeating some or all of steps 1-5.



IPG OVERVIEW

FIGURE 1

TP	-	Specifies processing of a specific input or output screen.
DBOP/DBSSA/DBRC	-	Three commands which define DLI data base operations including DLI calls (GU, etc.) SSA search criteria in summary or detail, and return code processing. Search criteria (SSA) may be specified in detail once and used by name in many DBOP commands.
MOVE/STORE	-	Two commands used to move data from one field or type storage to another. Moves literals and numbers to working storage fields.
CALCULATE	-	Specifies arithmetic calculations including simple or complex equations.
VALIDATE	-	Specifies validation of a working storage field in terms of another field(s) value, a given value, or complex criteria using arithmetic as well as and/or operators. If error detected, an output message and/or flagging of other screen fields may be specified.
GO/LABEL	-	Specifies branching within program.
IF	-	Specifies logical expressions to be evaluated as "true/false" conditions and branches.
LOOP	-	Specifies a sequence of commands to be executed a number of times.
QUIT	-	Specifies final processing for an input transaction including output of collected error messages, formatted screens or other "housekeeping".
OPLI	-	Provides freeform coding of PLI statements and comments.
SPA	-	Formatted layout and definition of fields for IMS Scratch Pad Area records.
SSA	-	Detail specification for uniquely labeled SSA's.

FIGURE 2 IPG PROCEDURE COMMANDS

Data Base Management Systems, D. A. Jardine (ed.).

IMPLEMENTATION TECHNIQUES FOR DATA STRUCTURE SETS

C. W. BACHMAN
Advanced Systems Technology Operation
Honeywell INformation Systems
Waltham, Massachusetts

1. INTRODUCTION

The application of sets to data structures has been described* and is the basis of several data base systems which provide the application programmer considerable freedom in navigating this way through a data base. These sets give a means by which the programmer can apply new access methods to a data base, and thus selectively process records based upon predeclared relationships. The new access methods* are listed in figure 1 below with three older methods to show the full range of choices which should be available to the programmer whether processing a single file or a data base.

Access Methods
Direct Access
Primary Data Key Access
Set Owner Access*
Set Member Access*
File Sequential Access

Figure 1

The combination of all five of these into a single base system is supported by the Honeywell Integrated Data Store (I-D-S) and the CODASYL Data Base Task Group Report. Index Sequential access packages, available from several manufacturers, make both the Data Key Access and File Sequential Access methods available to a programmer while working on the same file. (Primary Data Key Access is a generic name which describes both Randomized Access and Indexed Access which access a particular record based upon its primary data key.)

In the process of developing a particular data base system or of selecting one for use in developing an application system, there are two important factors to consider: functionality of the system and performance. The purpose of this paper is to briefly describe a number of set implementation techniques and then to examine relative performance of each with respect to specific set operations.

Section 2, describes 9 different set implementation techniques. For each of these basic forms, there are many possible variations; some of these variations are described below.

* Honeywell (1971), Dod (1966), CODASYL DBTG (1971), Bachman (1973, 1973 (a), 1973 (b))

Section 3 examines each of the 9 basic forms of implementation with regard to 14 different set manipulation functions.

2. SET IMPLEMENTATION TECHNIQUES

Sets have been implemented in many ways over the fifty or more years that effective information processing systems have been in use. These techniques have been selected for different reasons; hardware characteristics, particular processing requirements (i.e., batch vs transaction) and differing ratios of update vs. retrieval. The 9 different set implementation techniques listed below have all been used in one system or another and have successfully carried out some of the set operations which are part of the functionality of the data base set concept. No one technique is the best in all operations listed. Therefore, it can be expected that these basic set implementation techniques will survive and various modifications perfected. The listing below is simply a list and not an attempt to establish a taxonomy or classification system for such implementations.

1. Single Level Record Array: This form physically groups all the member records of a set into a table. All records are of the same format. The owner record is not in the table but has a pointer field which contains the address of the beginning of the table. A sorted magnetic tape with only member records is an equivalent form. In this case the first member record of each set occurrence is determined by scanning the tape for the first occurrence of the owner record's primary data key value which is stored in each member record as a secondary key. See Figure 2.
2. Multiple Level Record Array: This form merges an hierarchy of sets into a physically sequential array of records where the owner records are followed by their members. Each member in turn, if it is an owner of a set, will be followed by its own members, etc. A magnetic tape with merged master and detail records is an example of this form. A COBOL record (01 entry) with group or elementary items declared with an "OCCURS" clause is another example. Each in level (01, 02, 03) declaration with an "OCCURS" clause represents another level of sets in a hierarchy. In these cases, the multiply occurring COBOL item is equivalent to a member record (or fragment of one) by the definition of this article. See Figure 3.
   NOTE: Both forms of record arrays control the record's physical placement.
3. Pointer Array: In this form, the owner record has a pointer field which contains the address of an array of pointer fields. Each pointer field identifies a record which is a member of the set. Pointer fields generally contain the data base address of a record but may contain the primary data keys by which it may be located. See Figure 4.
4. Boolean Array: This form provides a Boolean (bit) array for each set. The array length in bits is to equal or greater than the count of the records in the file. Each owner record has a pointer field which contains the address of the set's bit array. Within the bit array, each bit represents a particular record in the file and is set to "1" if the record is inserted as a member, or set to "0" if it is not. See Figure 5.
5. Packed Boolean Array: This form condenses a bit array by replacing long strings of "0" bits with a flag which signals their absence. On large sparse arrays, compressions of 100:1 can be affected. See Figure 6.
   NOTE: Both forms of Boolean arrays prohibit the user from specifying any particular ordering of these sets. Records always retrieved is the sequence in which they are represented in the Boolean array.
6. List: This form provides the owner record and each member record with a pointer field which holds the data base key of the next record in the set. The last record holds a null value in the pointer field to signal the end of the set. See Figure 7.
7. Chain or Ring: This form is similar to the list form with the exception that the pointer field of the last member holds the data base key of the owner rather than a null value. Variations provide the owner record, or the owner

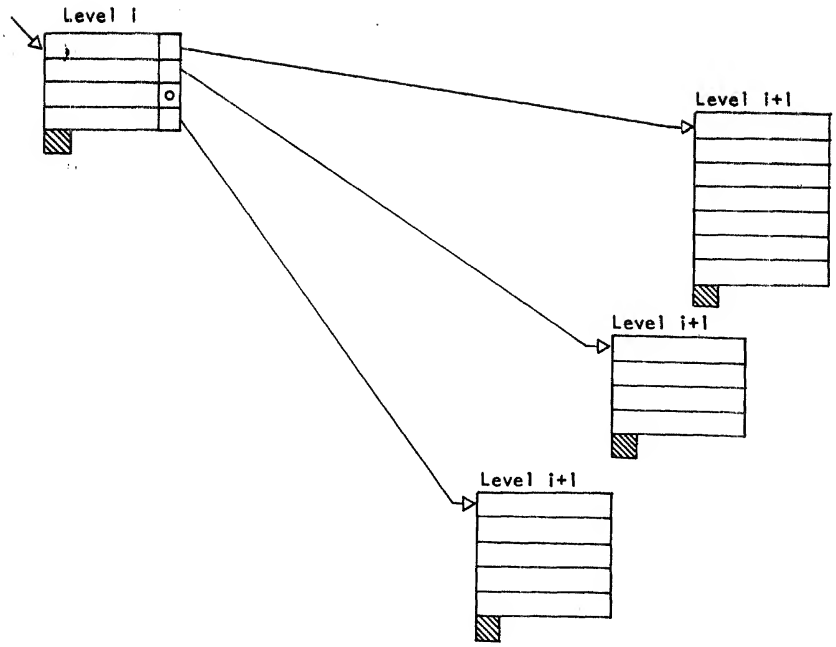


FIGURE 2 RECORD ARRAY

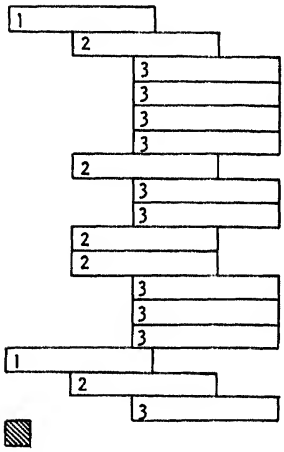


FIGURE 3 MULTI LEVEL RECORD ARRAY

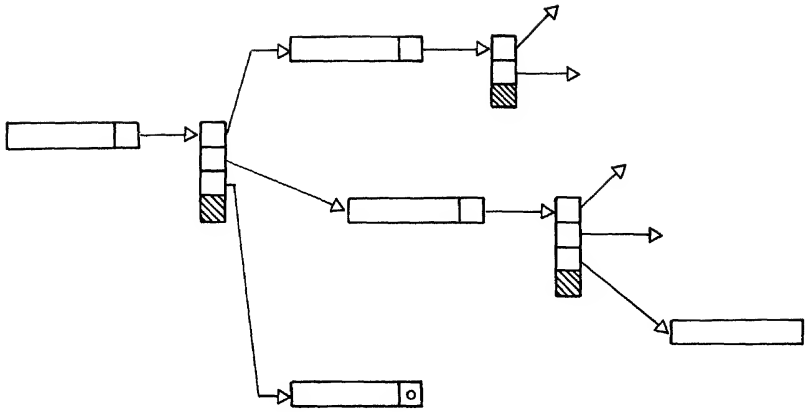


FIGURE 4 POINTER ARRAY

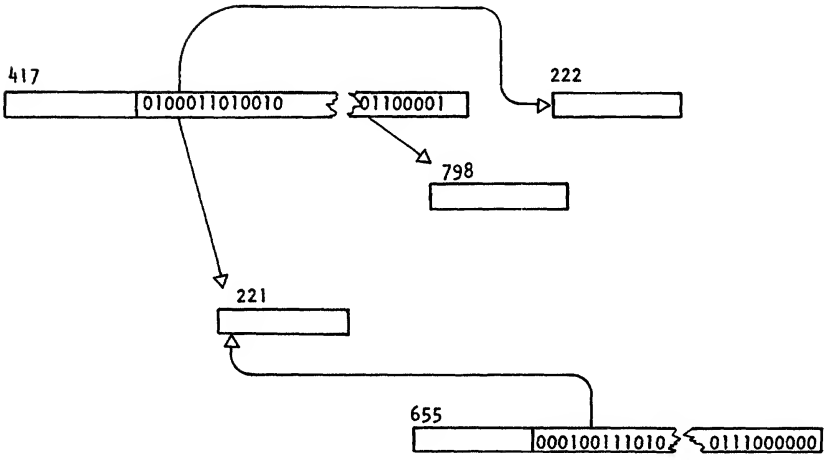


FIGURE 5 BOOLEAN ARRAY

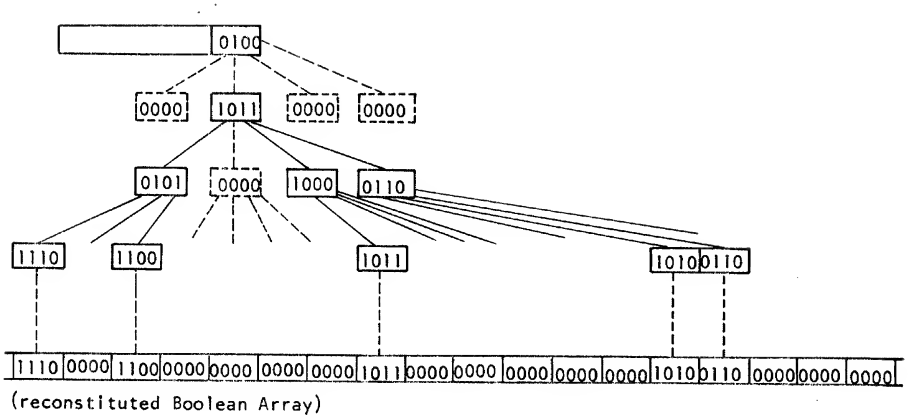


FIGURE 6 PACKED BOOLEAN ARRAY

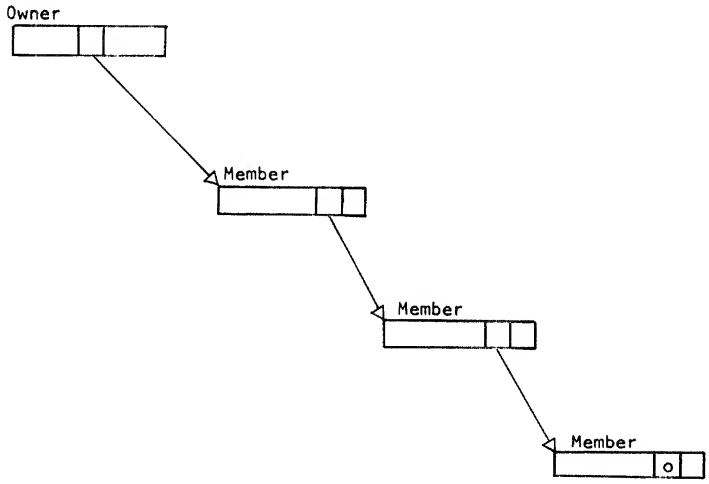


FIGURE 7 LIST

and all the member records with an additional pointer field to hold the address of the prior record. See Figure 8.

NOTE: Physical storage considerations allow a record in a list or chain to be "placed near" its owner record for one set, enhancing for the one set its processing proficiency when it is stored on a disc or drum.

8. Binary Tree: This form is specific to sets which are to be ordered in a data key value sequence. There are several variations on this implementation form. The least complicated form provides the owner record with one pointer field, and each member record with two pointer fields. The owner record points to the first member inserted into the set (or some other member if the tree has been balanced). The left pointer of each member points to one member and potentially a subtree of members all with lower data key values than the member doing the pointing. The right pointer points to one member and potentially to a subtree of members all with higher data key values. Both left and right pointers may be null. See Figure 9.
9. Phantom: This form provides each member record with a pointer field which points to the owner record. The owner record does not have a pointer to any of its members. It may optionally hold a counter which indicates the current number of members. This is used to prevent the owner from being deleted while there are still members in its set. The phantom form may be used alone, but is also used in conjunction with one of the other forms to implement the Set Owner Access method. See Figure 10.

3. COMPARATIVE PERFORMANCE CHARACTERISTICS

Table 1 gives the relative proficiency of each of the set implementation techniques against a list of fourteen different set manipulation operations. These figures are indicative of general proficiency and suggest where the usage of a particular form is good, bad or indifferent but should not be considered accurate enough to be used in a specific design without a detail study. In most applications, many different set operations will be used and the weighted effect of this mixture must be considered. Proficiency will be measured in terms of: excellent, good, fair, poor and terrible. These may be equated on a disc based system to: 1/3, 1, 3, 10, and 30 disc accesses per operation, respectively. Actual processor time has not been considered. Therefore, these figures should not be used to evaluate set implementation techniques for main memory structures. If there is no evaluation given, then that means that the operation is not supported by the particular implementation under the circumstances given.

The functions may be divided into three groups: set maintenance operations, set member access operations and the set owner access operations:

1. Set maintenance Operations:
   a) insert LIFO: This means to insert a new member as the first member of a set.
   b) insert FIFO: This means to insert a new member as the last member of a set.
   c) insert KEY: This means to insert a new member into a position within a set to preserve the data key sequence of themembers.
   d) remove: This means to remove a currently inserted record from a particular set in which it is a member.
2. Set Member Access Operations:
   a) find first: This means to access the first member of a set given the set owner identification. If the set is empty, return an empty set indication.
   b) find last: This means to access the last member of a set given the set owner identification. If the set is empty, return an empty set indication.
   c) Find i^{th}: This means to access the ith member of a set counting from the first member. If the set membership count is less than i, return an out-of-set indication.

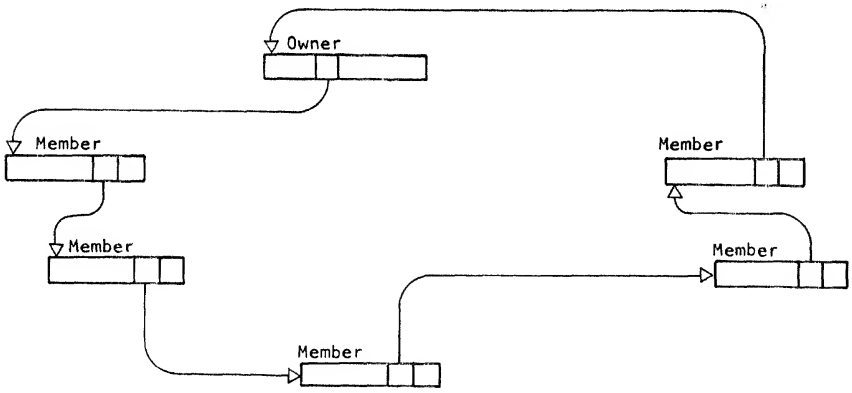


FIGURE 8 CHAIN/RING

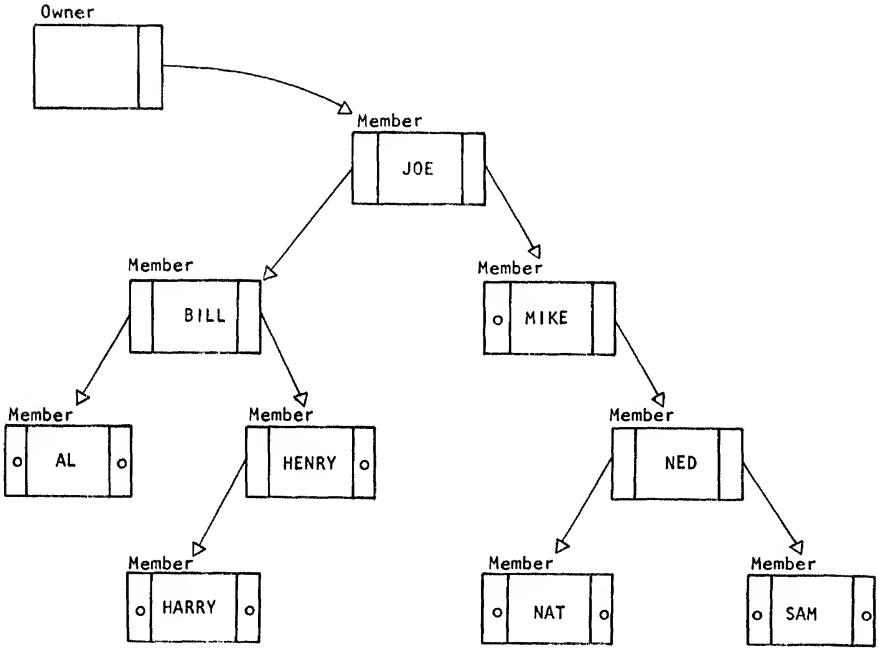


FIGURE 9 BINARY TREE

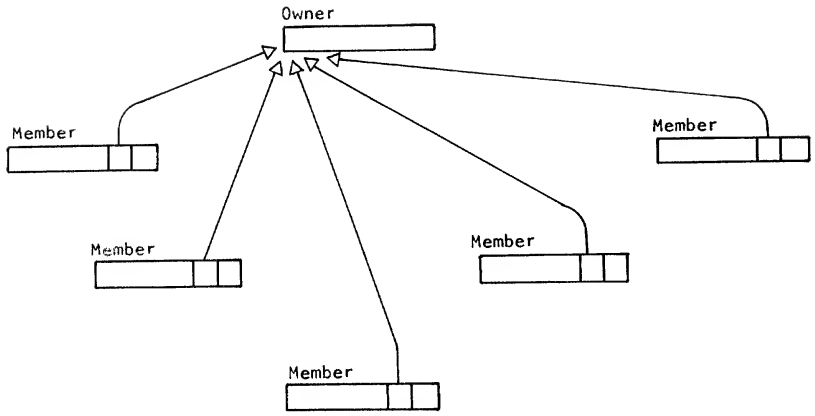


FIGURE 10 PHANTOM

SET IMPLEMENTATION TECHNIQUE EVALUATED FOR SPEED AGAINST SET OPERATIONS

	NSRT LIFO	NSRT FIFO	NSRT KEY	REMV	FIND FRST	FIND LAST	FIND I TH	FIND NEXT	FIND PRIR	FIND ALL	FIND KEY	FIND INTR	FIND SUM	FIND OWNR
Single Level Record Array.....	poor	fair	fair	poor	good	fair	good	excl	excl	good	fair	--	--	--
In file building sequence....	--	excl	excl	--	--	--	--	--	--	--	--	--	--	--
If deleted record left in place.....	--	--	--	good	--	--	--	--	--	--	--	--	--	--
Multiple Level Record Array...	terr	poor	terr	terr	fair	terr	poor	fair	fair	fair	terr	--	--	--
Bottom level only.....	--	fair	poor	--	excl	fair	good	excl	excl	excl	fair	--	--	--
In file building sequence....	--	excl	excl	--	--	--	--	--	--	--	--	--	--	--
If deleted record left in place.....	--	--	--	good	--	--	--	--	--	--	--	--	--	--
Pointer Array.....	good	good	poor	good	good	good	good	good	good	good	poor	good	good	--
Members placed near each other.....	good	good	fair	good	good	good	good	excl	excl	excl	excl	excl	excl	--
Boolean Array.....	--	fair	--	fair	fair	fair	fair	good	good	good	poor	good	fair	--
Packed Boolean Array.....	--	good	--	good	good	good	good	good	good	good	poor	good	good	--
List.....	good	terr	poor	good	good	terr	poor	good	--	good	poor	fair	good	--
Members placed near owner....	excl	fair	fair	excl	excl	fair	fair	excl	--	excl	fair	good	excl	--
Last member pointer in owner.	good	good	fair	fair	good	good	poor	good	--	good	fair	fair	good	--
Prior pointer in each record.	fair	fair	fair	fair	good	good	poor	good	good	good	fair	fair	good	--
Place near and prior pointer.	excl	fair	fair	excl	excl	good	fair	excl	excl	excl	fair	good	excl	--
In file building sequence....	--	excl	excl	--	--	--	--	--	--	--	--	--	--	--
Chain/Ring.....	good	terr	poor	good	good	terr	poor	good	terr	good	poor	fair	good	poor
Members placed near owner....	excl	fair	fair	excl	excl	fair	fair	excl	--	excl	fair	good	excl	fair
Last member pointer in owner.	good	good	fair	fair	good	good	poor	good	--	good	fair	fair	good	poor
Prior pointer in each record.	fair	fair	fair	fair	good	good	poor	good	good	good	fair	fair	good	poor
Place near and prior pointer.	excl	fair	fair	excl	excl	good	fair	excl	excl	excl	fair	good	excl	fair
In file building sequence....	--	excl	excl	--	--	--	--	--	--	--	--	--	--	--
Binary Tree.....	fair	fair	fair	fair	fair	fair	poor	fair	fair	fair	good	poor	good	--
Members placed near owner....	good	good	good	good	good	good	fair	good	good	good	good	fair	good	--
Phantom.....	good	good	--	good	--	--	--	--	--	--	--	--	--	good

TABLE 1

d) find next: This means to access the next member of a set given a currently inserted member. If the member given is the last member of the set, return and end-of-set indication.

e) find prior: This means to access the previous member in a set given a currently inserted member. If the member given is the first member of the set, return a first-of-set indication.

f) find all: This means to access all of the records in a set, one record at a time, in any arbitrary sequence. When the last member has been accessed, return an end-of-set indication.

g) find key: This means to access a particular record within a set based upon a data key value with the record. The set is assumed to be sorted. If there is no record with the specified data key value, then the next higher record is accessed and a missing-key indication is returned.

h) find intersection: This means to access all the records, one record at a time, which are currently inserted members of two specified sets. When the last dual member has been accessed, return an end-of-intersection indication.

i) find summation: This means to access all the records, one record at a time, which are currently inserted members of one or both of two specified sets. When the last member has been accessed, return an end-of-summation indication.

3. Set Owner Access Operations:

a) find owner: This means to access the owner of a set given the identification of any currently inserted member. If the member is not currently inserted into the specified set, return a not-inserted indication.

Frequently the performance of some operation, given a particular set implementation technique is seriously affected by some available option, or by some aspects of the way in which the operation is being carried out. The following notes are relevant to those special situations which were evaluated.

a) In file building sequence: This relates to evaluating the insert function where the insertion takes place as the file is first constructed and the insert operation is being carried out repeatedly on new members.

b) If deleted records are left in place: This relates to the logical removal of records which are physically in a record array to save compressing the array.

c) Bottom level only: This relates to the performance of operations on the bottom level of a record hierarchy only in contrast to all other levels.

d) Members placed near each other: This relates to an option which attempts to allocate physical space for all member records of a set within the same page or block such that there is a high probability that several member records in the same set will be physically accessed from secondary storage if any one member is accessed.

e) Members placed near owner: This relates to an option which attempted to allocate physical space for member records within the same page or block as the owner record such that there is a high probability that several member records in the same set will by physically accessed from secondary storage if the owner of any one member is accessed.

f) Last member pointer in owner: This relates to the option that the owner record has a pointer to the last member as well as to the first member.

g) Prior pointer in each record: This relates to the option that the owner record and all member records have a pointer to the prior record as well as to the next record.

h) Place near and prior pointer: This is a combination the "prior pointer" option and "members placed near owner" option.

EDITOR'S NOTE:

The paper orally presented by Charles Bachman was a working draft of the A. M. Turing Lecture which was to be presented at the ACM Annual Conference (August 28, 1973). It was published in the November 1973 issue of the Communications of the ACM. The comments which follow relate to the oral presentation and only indirectly to the paper on Implementation Techniques which immediately precedes.

DISCUSSION

CODD:

First comment: I want to publicly congratulate Charlie on his very fine prestigious award*. I feel that it couldn't have happened to a nicer person, a person who has played a very influential role in this field for many years. As evidence of this role we have the existence of a sizeable group of people, the CODASYL group, who have followed very closely the ideas of Charles Bachman. All this doesn't prevent me from disagreeing entirely with Charlie's thesis. I found his opening statement to be extremely provocative because he made the analogy between flat-land and flat-files. This is error number one. Let me trace some of the history that has been going on in this field. We began with punch card equipment that favored flat files: homogeneously formatted punched cards. We moved into the tape area and we started off with sequential files. Because tape units had problems of stopping and starting tape, the concept of blocking records came into existence and we had our first move towards a distinction between the logical view of data and the physical view of data. Now from that point we have been developing to a wider gap between the logical and the physical as we have tried to introduce device independence and now the more elaborate schemes of data independence. Newell, Shaw, and Simon and perhaps others introduced list processing ideas a long time ago, and they were very attractive to a certain portion of the programming population. There are programmers who love pointers and have fallen in love with them. The thing is that we now have an opportunity, because of the capabilities of today's equipment, to reverse the trend of making the logical view of data more and more complicated. In looking at the progress towards integration of files, we have noticed that the entities, previously processed separately, now have to be more and more heavily inter-related. Therefore, there has been developed in the whole thinking in this field an entire separation between the notion of an entity and the relation between two or more entities. This has resulted in the very elaborate data structure diagrams that we have seen displayed here this week. Now it so happens that a flat file and a collection of flat files has all of the power you want to deal with the n-dimensional world. You do not need to introduce any separate concept of relationships: the pointer-style relationships that we see with arrows on these diagrams. It so happens that you can consider the entities like parts, suppliers, and so forth and relationships like (so and so supplies such and such a part) in one way, a single way, namely the flat-file way. What are the advantages of doing this? By adopting a uniform flat-file approach you can simplify tremendously the set of operators needed to extract all the information that can possibly be extracted from a data base. Five or six simple operators can do this in the sense of being able to get all the permutations of the data (excluding transformations of the data). If you want to take statistics, of course you have to have statistical functions in addition to these operators. Moreover, if we

* C. W. Bachman was recipient of the Turing Award of the A.C.M. in 1973.

adopt logical views of the data that are based on flat files, we can bring these data bases into contact with casual users, including housewives. As I said in London recently, we may even get these data bases into contact with housewives before the concept of housewife becomes obsolete. Housewives can deal with shopping lists much better than they can deal with road maps. So the flat-land concept that you brought up I think introduces entirely the wrong notion in peoples' heads. It tends to say you have lost power if you have flat files; that you can by the addition of pointers introduce extra retrieval capabilities of some kind. This is not true. A flat file that is homogeneous and does not have duplicates happens to be what a mathematician calls a 'relation of degree n'. If you deal with your data in terms of relations of degree n, then the problem of growth becomes simpler because you do not make a distinction in the programming between relations of degree two and relations of degrees greater than two. When relations of degree two grow to become a relation of degree three you do not have to make changes in the way you program. Considering the fact the relations are obviously going to become a vital part of the technology of data base, and in view of the inevitable integration we are getting into, how many people here are familiar even with Bertrand Russell's relative product on binary relations, relations of degree two? I would say perhaps as many as 50%. How many people are familiar with operations on relations of higher degree? Probably less than 10% of the people here. I believe that this meeting is like a meeting of professional electrical engineers discussing circuit analysis or synthesis and 80% or 90% of them not being familiar with Ohm's Law.

BACHMAN:

I think Ted has some very excellent ideas. I can go within the walls of my building in Massachusetts and say "Those guys are Codd guys". There is this Codd school, and I think that is important. The most important thing we can do is to make sure that all good ideas get looked at and get compared very carefully for their merits and their disadvantages. I do thank Ted for his comments about the inevitability of integration and also about the aspects of relationships. I would caution him on his comments about pointers because I didn't mention pointers. I mentioned sets or relationships. Pointers are a way to implement those. In fact if you would look at the paper distributed, it discusses various sets of implementation techniques. One of the reasons that the paper was written is that a lot of us are dealing with a concept and have been dealing with them very successfully, if unconsciously for many years. You talk about an array or a table: those are particular implementations of sets. I am also a little concerned with the comments Ted made with respect to the fact that he was seeing more and more complex data structure diagrams. One of the problems we have is that only as they get more and more complex do they represent the real world we are dealing with. Therefore the better these map the world into our information systems the more responsive our information system will be to the real world requirements. That in the end is the service which we support.

METAXIDES:

I agree with Charlie (Bachman) that pointers are an implementation technique and they are completely transparent both in the declarations and in the users interface with the data base. As I see it, Charlie (Bachman) was not in any way saying that the one approach is more powerful than another.

Charlie (Bachman) requested feedback on the clarity of his presentation. In response I will try to summarize some of the major points he made to see if I followed his line of reasoning. Data base processing is a modelling operation and what we are modelling is the real world. The data base is a model of selective aspects of the real world, and insofar as it is a true model, the data base will be sturdy and can grow to reflect the changes in the real world as they occur. That could be done without impacting programs. The real world commonly has structure of arbitrary complexity. There is nothing that says the real

world is simple and straightforward. In fact one gets into the intricacies of developing an application, it is much more complex than it appears in the abstract. A data description language or a data base management system must therefore have structuring capabilities of great generality if it is to reflect the real world. Data structure provides navigation routes or road maps of the data base for programmers or non-programmers to find their way around this universe of the data base, which might be very large in scope. Commonality should be factored out, as it is in the real world. If in the real world there is only one occurrence of each department in a given division, then a faithful data base model will only have one occurrence of that department, and not n representations of it. That is what is reflected in the DBTG approach. Programs should have available to them not a specific technique for accessing the data base, but a package of all known techniques.

Is that a fair summary of some of the major points in your talk?

BACHMAN:

I think that's a very good job of picking out some of the points and bringing them forward again. The real world is a complex structure, but it is also a finite structure and not an infinitely variable structure. In looking at the data structure diagrams that were presented by the users on Monday, one happened to be a manufacturing control structure. Any of those who have been involved in manufacturing control systems and have some notion of the nature of information, and those who have been working with data structure diagrams, could immediately tell what that application was doing and not doing. That was an immensely powerful road map, if you like, of the n-dimensional data space they were treating. In a way our most critical problem in information systems is not communication with the computer but communicating with the people working around the computer and in the business. In this sense, data structure diagrams are extremely important in getting things done.

LEFKOVITS:

I find myself maybe a little unsteady with one foot on flat-land and one foot somewhere else, but I think there are two issues here. One is the data base as a model of the world and the other is, how do people see this model? I think we have a great variety of people and their particular needs are such that it is appropriate that they see the data in different ways. On one hand we are talking about an applications programmer who may be writing a program for a transaction. It is important that he should know a great deal about the details of how this data is really represented in the data base. On the other hand we have a housewife. It is inconceivable to me that the needs of these two people are in any way very similar, so I think there is a need for both views. If we look in the area of end-user facilities, I think we have that problem. We have different people with different problems and it happens to be appropriate that facilities should be made available such that different people can see the data in a manner which is more consistent with their way of thinking. We have a real challenge here, because one of the things that we are really striving for is the effective use of the computer. If we cannot use the computer itself to make life more convenient for people in the pursuit of their activities, then I think we have missed a good bet.

RABIN:

I was looking at Bachman's diagrams and also Codd's relations and I found that the diagrams are a reasonable way of representing the real world, and network of sets are a real way of getting close to the presentation of the real world in terms of the diagrams. However, from the user's point of view, the access is very procedural. At the same time, from an intellectual point of view, I was fascinated with Ted Codd's relational concept because of its mathematical beauty

in the sense that, you start with a set, the relation is a set and when you have a request, the answer is also a set. However, I tried to interface with some examples using relational calculus, and I found that one has to develop ingenuity in order to define such a request in terms of such a relational calculus language.

CODD:

It's true that mathematical relational concepts can be applied to any data base management system for obtaining a certain perspective on the system. You can discuss the kind of relations that are supported and the way they use various distinct techniques for supporting different kinds of relations. Very few systems in existence today, and they're only experimental ones that do it, support relations in a uniform way. But when you say that the IDS and DBTG concepts are in close correspondence with those you would use if you were to architect a system directly based upon a uniform approach to relations, I think you're definitely leading people astray. Let's go back to an example that was displayed by one of the speakers on the first day. I counted 13 varieties of Find. There's no need for more than one.

Now as to Jonas Rabin's comment. Different kinds of users do need, in my view, different kinds of languages. In this sense I do not understand Pat Nichols' single language notion. Programmers are going to continue to need programming-level languages, but we would like to step up their capabilities by introducing powerful relational operators such as projection, join, and division, which is not arithmetic division but something that corresponds, for example, to the action which is needed when you want to find those suppliers, each of whom supplies all parts supplied. The question also asked, how complex is it to frame queries in the relational calculus, which is a logic above Boolean logic? It's true that that kind of thinking is foreign to most programmers. Most programmers are familiar with Boolean logic, but not with the predicate calculus. I believe that we can protect programmers from having to become consciously aware of predicate calculus, but I think that they're going to be sucked into it because it's a very, very powerful tool, just as they've been sucked into Boolean logic, with no previous knowledge of it. Non-programming types will, for the most part, be provided with very job-oriented languages. I feel the only solution to people whose jobs are not involved with their interaction is interactive support which begins with a statement in free English. That I believe is within the state of the art today.

Data Base Management Systems, D. A. Jardine (ed.).

FUTURE TRENDS -- HARDWARE

L. C. HOBBS
Hobbs Associates, Inc.
Corona del Mar, California

1. INTRODUCTION

A modern computer system can be considered to consist of the following major types of equipment or subsystems:

- central processor
- storage
- communications
- input/output equipment
- terminals

Central processors and computer communications are each a subject in themselves which would require more time and space than can be covered in this paper. Input/output equipment is also a major subject in itself with many subcategories; it is also an area in which the progress being made is more evolutionary than dramatic or revolutionary in most cases. With the exception of replacements or alternatives for key-punch devices, the progress in input/output equipment in recent years has not had a major impact on the use or application of computer systems. Hence, central processors and communications systems are not covered in this paper; only an overview of input/output equipment is given; and primary attention is concentrated on memories and terminals. Significant progress is being made in memories and terminals from the standpoints of technology and cost, and their impact on system organization and utilization is significant.

The growing importance of mass memories and terminals, and the decreasing importance of main frame and conventional input/output equipment, in the computer market when viewed from a dollar volume basis, is illustrated by Figure 1 which presents estimates of the percentage of the total dollar volume of computer equipment represented for these categories in 1972. If the central processor is considered independently of main memory, then the dollar importance of the main frame is even less relative to the peripheral equipments. Separating the CPU and main memory is becoming a more valid distinction since more and more companies are designing computers in a way that permits the main memory to be more easily separated from the main frame to permit adapting more readily to new memory technologies. Also, other companies are supplying add-on memories which can be added to another manufacturer's computer.

These shifts in the relative dollar importance of different portions of the computer system are influenced by three major factors:

1. Advances in technology (particularly semiconductor technology) which have decreased the cost of logic, memory and other electronic portions of the system at a much more rapid rate than that for electromechanical peripheral equipments.
2. Changes in system design approaches, such as multi-user or multi-access systems, distributed processing, virtual storage, multi-processor systems, etc.
3. Changes in user requirements and application approaches, particularly greater on-line use of computers and increased use of communications oriented remote terminal systems.

The impact of changes in technology and user requirements on input/output memory, and terminal equipment requirements is reflected by several trends, including the following:

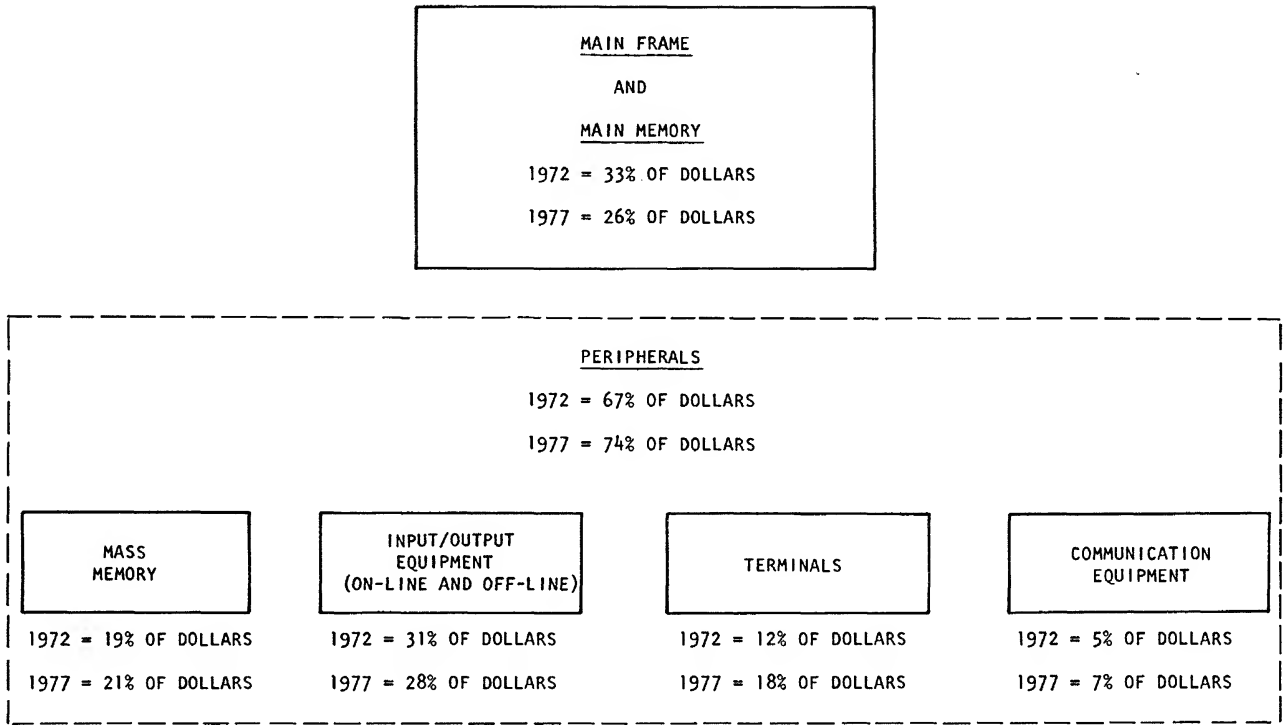

MAIN FRAME
AND
MAIN MEMORY
1972 = 33% OF DOLLARS
1977 = 26% OF DOLLARS
PERIPHERALS
1972 = 67% OF DOLLARS
1977 = 74% OF DOLLARS
MASS MEMORY
1972 = 19% OF DOLLARS
1977 = 21% OF DOLLARS
INPUT/OUTPUT EQUIPMENT (ON-LINE AND OFF-LINE)
1972 = 31% OF DOLLARS
1977 = 28% OF DOLLARS
TERMINALS
1972 = 12% OF DOLLARS
1977 = 18% OF DOLLARS
COMMUNICATION EQUIPMENT
1972 = 5% OF DOLLARS
1977 = 7% OF DOLLARS

FIGURE 1

1. Acceleration of the trend toward on-line communications oriented systems with a consequent increase in the requirements for terminals and communication processors and reduction in the growth of requirements for certain types of input/output equipments.
2. Increased on-line entry of data via CRT terminals and keyboard printer terminals as direct replacement for keypunch devices.
3. Trends toward source data collection using CRT terminals, keyboard printer terminals, point of sale recorders, factory data collection terminals and similar devices at the point of origin of the data, which eliminates keypunch devices for subsequent data entry.
4. Development of distributed processing systems which utilize "smart" or "intelligent" terminals that include processing capability in the terminal, including up to 65,000 bytes of storage.
5. Trends in business applications toward combined small business machine and smart terminal equipment in which the local equipment (e.g., in a warehouse or district sales office) operates as a stand-alone small business system with its own local data base during the day, but acts as a terminal at night to be polled by a remote central computer over telephone lines to collect the results of the day's data entry and processing operations and to update the data base in the local system.

2. INPUT/OUTPUT EQUIPMENT

In considering input/output equipment, it is important to clearly distinguish between two categories of equipment which have basically different functions but which overlap to some extent in the use of specific equipments. The first category, which can be called "true input/output" equipment, includes equipment used to transcribe or convert data from some other form or media into computer readable digital data or vice versa. This includes:

Keyboards
Key punches (key to card)
Key-to-tape devices
Key-to-disc devices
Audio response units
Printers
Plotters
Graphic input devices
OCR, MICR, and mark sense equipment
COM and CIM equipment

Although these devices may operate directly into or out of the computer, they are frequently used with some form of intermediate storage in an off-line mode.

The second category consists of equipments which are connected directly to the computer. They appear as input/output devices to the computer itself, but they may not appear to be "input/output" from an overall systems standpoint. These include:

Punched card readers
Punched card punches
Punched paper tape punches
Punched paper tape readers
Magnetic tape units
Badge readers

The overlap is indicated by the fact that a magnetic tape unti may be used also as the output for a key-to-tape device or for the input to an off-line printer.

Punched Card Equipment

Punched card equipment has evolved slowly except for the introduction of 96-column card equipment a few years ago with the initial introduction of the IBM System 3. Recently, 96-column punched card equipment has been added to the product line by several other computer manufacturers and it now offers an alter-

native to the conventional 80-column punched card. Although 96-column punched cards provide smaller physical size and lower cost handling equipment, the 80-column card remains dominant because of its entrenched position.

Data Entry Equipment

Key input devices offered as an alternative to conventional card keypunches are one of the most important recent developments in input/output equipment. These include key-to-tape devices, key-to-cassette devices, key-to-disc systems, and key-to-diskette devices. Two relatively new developments that may have a great impact on future systems are the key-to-diskette system announced last year by IBM and the combination data entry system offered by several companies in which OCR readers are combined with key-to-tape or key-to-disc systems. On-line terminals and source-data collection are also having an impact on the use of conventional keypunch equipment.

The different approaches to data entry can be classified and summarized as follows:

- Off-line Data Preparation
  - Keypunch
    - Unbuffered
    - Buffered
  - Key-to-Tape
    - IBM Compatible
    - Cassette
  - Key-to-Disc
    - Shared Processor
    - Diskette
  - Other
    - MICR
    - OCR
    - OMR
    - Etc.
- Direct On-Line Data Entry
  - Data entry terminals
    - Keyboard/printer
    - CRT
  - User terminals (data collection)
    - Keyboard/printer
    - CRT
    - Smart Terminals
    - Etc.
- Source-Data-Entry Terminals
  - Point-of-Sale Recorders
  - Credit-Verification Terminals
  - Factory-Data-Collection Terminals
  - Etc.

The same type of operator terminal may be used in more than one of the categories above. For example, a very similar CRT terminal may be used as an operator station in a large key-to-disc system, as a direct on-line data entry device at the computer site, or as a remote terminal for data collection by a user (i.e., one who is using the terminal for functional purposes rather than merely to transcribe data).

Data collection applications differ from other data entry applications in that the data is collected at the source and the terminal is oriented more toward the operator or user, in contrast to conventional data entry applications in which the operator primarily transcribes data in the same way that a keypunch operator does. User data collection as a form of direct on-line data entry differs from source data entry in that the user operates with a general purpose type terminal (e.g., CRT, keyboard/printer, etc.) to enter text or fill in blanks in forms; while in source data entry, a specialized (usually) terminal is used to capture data by means other than (but including) operator keystrokes--e.g., by reading badges, merchandise tags, credit cards, etc. User data collection includes two

different modes of operation. The first is a pure data entry operation in which the terminal may provide assistance to the operator (e.g., formats, validation, etc.) but in which there is no extensive interaction between the operator and the computer. In the second, the terminal may again assist the operator in the data input function, but in addition there is interaction between the operator and the computer in which the data entered, or the form in which it is entered, depending upon a two way conversation between the operator and the computer.

There is a strong trend toward direct on-line entry in contrast to off-line data preparation, but at present off-line data preparation still represents the dominant portion of the market. However, by 1980 direct on-line data entry and source data entry are expected to dominate.

Printers

The major developments in printers are those providing lower cost rather than higher performance. The exception to this is found in character serial printers where several of the newer units offer speeds up to 300 characters per second in contrast to the 10 to 30 character per second rates of a few years ago. Even this, in many cases, is a cost consideration since the higher speeds permit character serial printers to be used instead of more expensive line printers in many applications. These advances in speed of character serial printers have been made primarily by using a dot matrix impact printer technique or non-impact printer techniques (e.g., electrostatic, magnetic, thermal, or ink spray) rather than the conventional print hammer impact techniques found in slower character serial printers. These new character serial printer techniques are having a major influence on minicomputer systems as well as on terminals. The major technological advance in impact line printers has been the use of flex-pivoted hammers actuated by a voice coil mechanism. This approach is also found in some of the new character serial printers.

OCR, OMR, and MICR

The major progress in recent years in OCR has been in cost reduction and in the combining of relatively low cost OCR page reading devices with key-to-disc systems. Optical mark reading (OMR) is coming into wider use as an input means for unskilled personnel such as clerks and meter readers. Examples include a point-of-sale terminal for fast food chains in which the clerk marks the order on a preprinted OMR card and medical applications in which the doctor marks a preprinted card for blood tests. Magnetic ink character recognition (MICR) is still used almost exclusively for check coding and banking applications.

Badge Readers

The term "badge reader" is a generic term used to indicate devices which can automatically read encoded information from identification badges, credit cards, merchandise tags, and similar types of encoded cards. The primary uses of badge readers are in point-of-sale, credit verification and factory-data-collection terminals, which are discussed later. Although many techniques have been developed for encoding cards or badges, the three most widely used ones are punched holes, magnetically encoded strips, or optical marks. There is some speculation that OCR techniques may be used more in the future for reading printed characters on cards or badges, but this is not widespread at present except in the case of oil company credit cards. In the oil industry at present the imprinted sales slip is later read by an OCR device. However, there is increasing interest in reading the embossed characters on the oil company credit card directly in credit verification terminals at the point of sale in gasoline service stations. Magnetic strip encoding is now widely used on bank credit cards and on travel and entertainment credit cards, and in some cash dispensing systems. Both the International Air Transport Association (IATA) and the American Banking Association (ABA) have prepared magnetic strip encoding standards and most magnetic strip cards are now designed to accommodate either standard. The primary difference in the two standards is the encoding density.

Magnetic Tape Units and Cassettes

Advances in magnetic tape technology have taken two directions--increased density and data rates for high performance industry compatible tape drives, and higher reliability and lower cost for magnetic tape cassette and cartridge devices. The recording densities of conventional industry compatible high performance tape drives have been increased from 1600 bits per inch to 6250 bits per inch and data rates have been increased to as high as 1,250,000 characters/second. The use of magnetic tape cassette devices has been enhanced by higher reliability, lower cost and the recent agreement between American National Standards Institute (ANSI) and the European Computer Manufacturers (ECMA) on a standard for cassettes and data recording techniques.

Despite the widespread use of the Philips type cassette, other companies are still pushing the development and application of higher performance magnetic cartridge devices. Although magnetic tape cassette devices are clearly in the lead at this time, it is difficult to predict the impact that some of the magnetic tape cartridge devices will have on this market. This uncertainty is further compounded by the introduction of "floppy" magnetic discs which provide an alternative to the use of magnetic tape cassettes or cartridges in many applications. The "floppy" discus are somewhat more expensive, but they can provide semi-random access and their manufacturers claim lower error rates. Tape cassettes, tape cartridges, and "floppy" discs are widely used in terminals and data entry devices.

COM and CIM

Computer-output-on-microfilm (COM) and computer-input-from-microfilm (CIM) have continued to be market disappointments but many of their proponents remain optimistic. State-of-the-art COM devices permit alphanumeric reports to be recorded at speeds ten to twenty times that for electromechanical printers and reduced up to 48 times to microfiche size. COM is becoming increasingly important in the photo-composition market. In addition to speed and document size COM devices offer advantages in their ability to intermix alphanumerics and graphics.

3. MEMORIES

Hierarchies of storage devices have been used in computer systems since the earliest days of the computer industry. The systems designer would like a very large capacity, very high speed, and very low cost memory, but unfortunately these three characteristics are not found in the same device. The use of a hierarchy of storage devices to combine the more attractive features of different types of storage technologies permits the system designer to achieve a more reasonable balance between capacity, speed and cost. The designer attempts to use each different type of storage device in a way that minimizes its disadvantages and maximizes its advantages.

The functional types of storage found in such hierarchies include:

Discrete bit storage: for temporary storage of indicators, control signals, and results of previous operations.

Registers: for temporary storage of data and instructions, for implementing arithmetic and logical operations, and for addressing larger memories.

High speed control or scratch pad memories: for multiple arithmetic or control registers and for temporary storage of intermediate results, frequently used data, constants, and short subroutines which are being iterated.

Main internal memories: for storing data and instructions of the program currently being executed and alternate programs that may be needed rapidly in the event of a real-time interrupt.

Solid-state auxiliary on-line storage: for storing frequently used tables, data, and alternate programs that may need to be called into main memory rapidly.

Electromechanical auxiliary on-line storage: for storing large data files

and programs that may be called by the computer but where speed is less important and capacity and cost are more important than for solid-state auxiliary storage.

Off-line auxiliary storage: for storing very large data files and large program libraries which are used sufficiently infrequently to permit a manual operation to make the storage available to the system.

There are, of course, many other uses for most of the levels of storage listed above but the ones stated are typical of their major functional uses. All of these levels are not necessarily found in a particular computer. Each of these categories or functional types of storage require different combinations of speed, capacity, cost, and other characteristics. A specific memory technology may overlap two or more categories, but the relative importance of the different characteristics, and, to some extent the design approaches and criteria, vary from one category to another.

The auxiliary storage categories represent very large capacity bulk storage that is usually addressed by the computer in large blocks rather than by individual words. The average access time (except for solid-state mass memories) is usually one or more orders of magnitude slower than that of the high speed internal memory. On-line auxiliary storage is directly available to the computer under computer control without manual intervention. Off-line auxiliary storage normally requires a manual operation to place the storage media on the read/write device (e.g., a magnetic tape unit) that is controlled by the computer.

Most off-line storage equipments, such as magnetic tape units, punched paper tape units, and punched card units, are commonly classed as input/output equipment since they act as input/output devices to the central processor. Unfortunately, this tends to obscure the fact that these devices are actually service an off-line auxiliary storage function in the overall system rather than an input/output function. They should not be confused with "true input" and "true output" devices such as keyboards and printers.

Some types of mass memories, such as magnetic disc files with removable disc stacks, are on-line auxiliary storage devices with respect to the cards or discs actually on the device at a given time. However, they act as read/write and access mechanisms for off-line storage with respect to stacks of discs on a shelf which have been written by a device previously and will be read by a device subsequently.

The two levels in the storage hierarchy of primary importance, which will be covered further in this paper, are the main internal memory and electromechanical on-line auxiliary storage. Semiconductor devices have already taken over the discrete bit storage, register, buffer, and high speed scratch pad or cache memory functions. For example semiconductor memories are already used as buffers and refresh memories in over 40 terminals. The primary off-line auxiliary storage media is magnetic tape which was discussed as an input/output device in the preceding section. Solid-state on-line auxiliary storage applications are disappearing because of the constantly increasing capacity and decreasing cost of main memory and the reduced price and higher speed of small head-per-tract discs and drums. As new technologies, such as magnetic bubble memories or semiconductor charge coupled devices (CCD) memories are developed with significantly lower costs per bit, the solid-state on-line auxiliary storage may again be utilized in future systems designs. This is an illustration of the important fact that the design of the storage hierarchy and the characteristics of available memory technologies are closely interrelated. The advent of a new memory technology may eliminate one of the levels in the storage hierarchy by combining two functions into a single memory or may create a new level in the storage hierarchy by providing performance and price combinations intermediate between those of two of the existing levels in the storage hierarchy.

Main Internal Memory

The performance of main memory has been increasing and the price decreasing consistently since the introduction of the first commercial computer in 1951. Univac I had 1000 words of memory, and programmers were elated when Univac II was introduced which doubled the main memory to 2000 words. Less than 20 years later

anyone who does not have a half million words of main memory is hardly considered in the big leagues. The improvements to date in speed, capacity and cost have been almost entirely a function of technology advances in magnetic core memories which has been the dominant main memory technology for almost 20 years. During this period magnetic core memory cycle times have been reduced from approximately 10 microseconds to less than 500 nanoseconds.

Many have predicted the demise of the core memory periodically during the past ten years as different new memory technologies have been announced, such as planar magnetic thin-film memories, plated wire memories, flute memories, monolithic ferrite memories, cryogenic memories, and others. All of these alternative memory technologies proposed in the past failed to replace the magnetic core memory because of the "moving target" phenomena. Those developing a new memory technology would propose cost and specifications superior to those of core memories only to find that, before their devices were ready for the marketplace, core memory technology had advanced to the point that the price and performance of core memories were superior to those of the new technology. Core memories have maintained their dominant position because of the suppliers' ability to continuously decrease the cost per bit and increase the speed and capacity. None of the new technologies were able to achieve a sufficiently large production volume base to permit cost competition with core memories.

For example, plated wire memories may have been able to compete with core memories given an adequate production volume, but the cost and processing technology development never permitted them to achieve an adequately large production volume. This in turn made it impossible to achieve the necessary cost objectives to displace the established core technology and production facilities. Semiconductor memories on the other hand can be processed in existing semiconductor facilities which are being continuously expanded to meet requirements for other types of semiconductor devices. The total semiconductor memory requirements at the package level in 1975 will represent less than 10% of the total semiconductor market. Providing capacity for memories is a relatively small incremental change for the semiconductor industry in contrast to the establishment of unique large volume fabrication and test facilities that would have been required for any of the other technologies that have been proposed as alternatives to magnetic core memories.

For the first time magnetic core memories face a serious threat from another technology which is itself experiencing rapid cost decreases and performance increases. Semiconductor memories offer an alternative technology that can compete not only with today's magnetic core memory price and performance but with those anticipated for the future. In fact, semiconductor memories enjoy the advantage that their "learning curve" is still declining rapidly while that for magnetic core memories is tending to level off. Semiconductor memories also enjoy an advantage that was not present for any of the previous competitive memory technologies all of which had to succeed as a memory device or not at all. Research and development in semiconductor technology and expansion of semiconductor processing facilities is required and will occur for other semiconductor applications, even if semiconductor memories were not developed. Hence, semiconductor memories benefit from overall advances in semiconductor technology and are built on top of an existing production base. Hence, for the first time a new memory technology can be produced in a facility that has sufficient volume to permit competition with the established core memory technology on an equal basis.

Semiconductor memories present the first real thread to the dominance of magnetic core memories for main internal memory applications as indicated by the fact that over 30 new computers have been introduced with semiconductor main memories by over 20 different companies. The leading computer supplier has demonstrated a complete commitment to semiconductor memories in all new machines, including at least six major computer product lines and one communications processor. For most applications today requiring large capacity storage (e.g., in excess of 500,000 bits) magnetic core memories are still competitive and probably offer some cost advantage over semiconductor memories. However, for capacities below 250,000 bits semiconductor memories are more cost effective at this time.

Since the magnetic core memory is a reliable proven device that is well established, it will continue to be used in existing computer designs for several years. However as the cost of semiconductor memories continues to decrease, particularly with the availability of the 4096 bit NMOS semiconductor memory package in production quantities in 1974, new computer designs will switch more and more to semiconductor memories to achieve cost and speed advantages.

There are four major semiconductor technologies that can be utilized for memory applications:

Bipolar
P Channel MOS (PMOS)
N Channel MOS (NMOS)
Complimentary MOS (CMOS)

Bipolar semiconductor memories offer higher speed but are also more expensive. PMOS memories are slow with speeds roughly comparable to, or somewhat slower than, those for current state-of-the-art magnetic core memories; but the cost of PMOS memories is significantly less than that for bipolar memories. PMOS is the major semiconductor technology in terms of numbers of bits delivered in 1973, although bipolar memories are somewhat ahead in dollar value because of the higher price per bit. NMOS memories have been developed more recently with sample quantities now being delivered and large volume production deliveries anticipated in 1974. NMOS memories can provide both higher speed and lower cost than PMOS memories because of the higher bit density on the semiconductor chip. Although NMOS memories are not quite as fast as bipolar memories, they are much lower cost and are closer to the speed of bipolar memories than is the case for PMOS memories. Hence, NMOS memories are expected to become the dominant semiconductor memory technology and, in fact, the dominant technology for buffer and main memory applications by 1975. CMOS memories are not expected to compete with PMOS and NMOS for general purpose buffer and main memory applications, since they are more expensive than PMOS and NMOS and slower than bipolar memories. However, they will find applications in special cases because they are easier to design into a system, are compatible with CMOS logic, and require lower power. CMOS memories will be used in applications where capacities are small, environmental conditions are difficult, and low power is required.

The most widely used semiconductor memory device from independent suppliers today is the 1024 bit PMOS package, but the 4096 bit NMOS package is expected to come into widespread use in 1974. By 1977 or 1978 a 16,000 bit NMOS package is anticipated. The price of bipolar semiconductor memory packages (at the component level) is expected to decrease from approximately 1.5¢ per bit in 1973 to less than 0.5¢ per bit in 1977; the price for PMOS memories from less than 0.5¢ per bit in 1973 to less than 0.3¢ per bit in 1977; and the price for NMOS memories from 0.5¢ per bit in 1973 to less than 0.2¢ per bit in 1977. The computer system user must be careful not to be mislead by these prices which are intended to illustrate the trend. These prices are for semiconductor packages, but systems costs will be in the order of twice as high and end-user prices in a computer system may be several times the package prices quoted here. Access times will be in the 30 to 300 nanosecond range, depending upon the semiconductor technology used and the trade-offs made between speed and cost.

Auxiliary Storage

From the standpoint of Data Base Management Systems the mass memory is of primary interest. As pointed out earlier, solid-state on-line auxiliary storage has decreased in importance and use because of the lower cost and increased capacity of the main internal memory. On the other hand off-line auxiliary storage, such as magnetic tapes, do not provide the direct computer access usually required in a data management system. Hence, on-line electromechanical auxiliary storage and alternative future technologies are of primary importance. The dominant on-line auxiliary storage technology today and for the foreseeable future is the magnetic disc file.

Electromechanical mass memory can be broken into three categories. The first is fast head-per-track disc or drum storage providing one to five million bytes

of fast access (i.e., five to 10 milliseconds) storage for swapping discs and real-time applications. The second is low cost (i.e., $3,000 to $5,000) moving-head disc storage with capacities in the five to twenty-five million byte range and access times in the 50 millisecond range (average including latency and head positioning). The third is large capacity moving-head disc files priced in the $8,000 to $16,000 range with capacities from 50 million to 200 million bytes and access times in the order of 30 milliseconds. These prices are on an OEM rather than end-user basis since end-user prices vary widely depending upon the system manufacturer's price and mark-up structure. Other electromechanical mass memories that have been used on some systems, such as magnetic tape loops and magnetic card units, are not a serious consideration at this time, but one large manufacturer is reported to be preparing the introduction of a large capacity mass memory using up to 10,000 tape cartridges.

For data mangement systems, the type disc of primary interest is the large capacity moving head disc. Although a few companies now offer a double density version of the IBM 2314 type disc drive with 49 million bytes capacity, the major type of disc drive today for large capacity systems is the IBM 3330 disc drive and similar units manufactured by other companies. This drive provides a capacity of 100 million bytes, an average latency time of 8.3 milliseconds, an average access time of 30 milliseconds, and a data transfer rate of 806,000 bytes per second. Several companies are known to be developing a double track density version of this type device which would provide 200 million bytes capacity with the same transfer rate and approximately the same access times. These devices are expected to be available in the very near future. Disc drives with capacities up to 400 million bytes per drive are anticipated in 1975 or 1976. The trends in magnetic disc recording technology and reasonable expectations for the future are summarized below:

	IBM 3330	1974 to 1975	Post 1975
Bits/inch	4400	6000 to 8000	10,000
Tracks/inch	200	400 to 600	800
RPM	3600	3600	4,800
Average Access (MS)	30	30	20
Megabits/sec	7.5	10 to 15	25

One of the major recent developments that have been provided in his density magnetic disc drives are improvements in the magnetic coding tecynology and the use of error correction techniques in the controller to permit the utilization of narrower tracks and higher data rates. One of the major advances in some disc drives now being introduced and anticipated for the future is the development of a positioning system in which the head is servoed to the track being read rather than using a separate set of servo tracks on an end disc. This permits much higher track density and hence capacity but requires a more sophisticated addressing and servo system.

There are no new technologies which will be competitive with large capacity discs in the foreseeable future for computer mass storage. Magnetic bubble memories have been highly touted for several years, particularly as a replacement for high speed discs and drums. Since bubble memories are relatively low speed and must be operated in a serial mode, it is unlikely that they will compete with moving head disc files. Also, the future of the bubble memory is becoming more questionable since some of the early claims have yet to be realized, and the newer charge coupled devices (CCD's) appear to offer most of the advantages of bubble memories plus higher speed and lower cost. Both bubble memories and charge coupled devices are more likely to compete with fast access head per track discs and drums than with large capacity moving head disc files. Both of these devices are at least three years or more away from practical commercial applications.

Optical memories have been touted for many years, and the advent of lasers have accelerated work on optical memories. However, there are many materials and technical problems to be solved. There is no reason to believe that optical memories will impact the disc market before the late 1970's, if then. However

work on optical memories is continuing and it is reasonable to expect that optical techniques or some other new technology will eventually supercede the conventional magnetic surface recording as this technology approaches a plateau. However, it must be emphasized that this should not be anticipated in the near future.

4. TERMINALS

Among the many changes that have taken place during the past few years in computer systems design and applications, perhaps the most important is the increasing trend toward the use of on-line systems that involve man-machine interaction and communications between the computer site and a remote user. This trend has given impetus to the design of terminals to facilitate the use of the computer by the non-specialist. On-line computer based systems and specialized terminals designed to meet the requirements of a specific type of user are becoming an integral part of the day to day operation in such diverse fields as retail sales, manufacturing operations, airline reservations, and engineering design.

Typical applications which lead to the increase in the use of on-line systems include:

Inquiry systems (e.g., stock quotations, airline reservations, etc.)
Transaction systems (e.g., point-of-sale, factory data collection, etc.)
Time sharing systems
Instrumentation and control systems
Automated test systems
Communications systems

Some of these applications are well served by general purpose terminals, such as teleprinters or alphanumeric CRT terminals, while others require special purpose terminals which are designed specifically to meet the requirements of a particular application. A primary example of this is the point-of-sale and/or credit verification terminal designed for use in department stores and other retail sale applications.

The rapid decreases in the cost of computer logic and storage elements in conjunction with the relatively constant cost of communications facilities are making it increasingly attractive to perform as much of the processing as possible at the terminal site in order to minimize the data transmission requirements and costs. Hence "smart" or "intelligent" terminals which include stored program processing capabilities in the terminal are being used increasingly. This leads to a form of distributed processing system in which part of the work is done locally in the terminal and part at the central computer site.

Although remote terminals are addressed primarily here, much of the discussion from a technical viewpoint is also applicable to local terminals which may be connected directly to a computer without the use of outside communication facilities. Remote terminals connect a user, who is remotely located, with a computer by means of communications provided by telephone lines or some type of high speed communications facilities. The proper development of remote terminals, communication facilities, and computer hardware and software capabilities will bring computer functions to almost every large and small plant, office, and store--perhaps eventually to the home.

Subsets of The Remote Terminal Market

The remote terminal market includes those used as time sharing terminals, inquiry terminals, remote-batch processing terminals, point-of-sale and credit-verification terminals, factory data collection terminals, and miscellaneous types. Time-sharing terminals include alphanumeric terminals (teletype, electric typewriter, CRT/keyboard, etc.), graphic terminals (plotters, CRT with graphics input, etc.) and smart terminals. From an equipment standpoint there is some overlap among the different categories of terminal applications cited above. For example, an alphanumeric terminal (e.g., a Teletype with paper tape reader and punch) may be used either as a time-sharing terminal or as a slow remote-batch terminal. Hence for purposes of analysis in this paper, remote terminals will be

broken into the following subsets:

- Keyboard/printer terminals
- CRT terminals
- Smart terminals
- Remote-batch terminals
- Point-of-sale and credit-verification terminals
- Factory data collection terminals
- Miscellaneous terminals

All of the above terminal types are discussed briefly in the remainder of this section, with the exception of "smart" terminals and point-of-sale and credit-verification terminals, which are discussed in greater detail in subsequent sections.

Keyboard/Printer Terminals

Keyboard/printer terminals provide a hard copy which is essential in many applications but these terminals are slow, have high maintenance costs, and are usually quite noisy. The most common and widespread example of keyboard/printer terminals are the hundreds of thousands of Model 33 Teletypes presently in use. Other keyboard/printer terminals are designed around electric typewriters. Several companies have designed newer types of keyboard/printer terminals with different print mechanisms, either to serve specialized applications or to overcome some of the present disadvantages of Teletype or electric typewriter terminals.

More sophisticated terminals designed for heavy-duty usage, faster transmission rates to and from the computer, upper-and lower-case characters, and more extensive editing and formatting features are also available. The more sophisticated terminals sometimes include an electronic or magnetic buffer memory in the terminal capable of holding one line of data. This permits faster transmission speed to and from the computer for these terminals which are sometimes called "buffered typewriter terminals".

In interactive problem-solving applications (e.g., conventional time-sharing applications), hard copy is usually required and low cost is very important. Hence, keyboard/printer terminals dominate these applications at present. Although the speed of keyboard/printer terminals is a disadvantage, it is not as serious in these applications, since the user frequently stops for extended periods of time to think and consider what to do next, between entering and receiving relatively limited amounts of data. An ideal combination for this type of application is a keyboard/printer terminal with an optional CRT display. With this combination the keyboard CRT can be used in intermediate interactive operations to gain the speed of the CRT display. The printer can be used to obtain a hard copy of the final results after the user has reached the point where he wants a permanent copy.

The major advances in keyboard/printer technology are in new types of printer mechanisms--non-impact printers (which are quieter, more reliable, and of a higher speed) and improved impact character serial printers with speeds up to 120 characters/second.

CRT Terminals

CRT terminals are used for both alphanumeric and graphic applications, i.e., the display of text alone and the display of mixtures of text and diagrams. Much higher performance is required in most graphic terminals with consequent higher costs. A wide variety of alphanumeric CRT terminals are available at present. They have in common the ability to display alphanumeric information in lines on the fact of a CRT, to accept information from a remote source such as a computer, to enter information from a manual keyboard, to edit and rearrange data, and to transmit data to remote destinations. A small buffer memory is usually provided to permit refreshing the information on the face of the CRT at a sufficiently fast rate to avoid noticeable flicker. In some cases this refreshing may be handled directly from the computer's internal memory, but this represents a heavy burden on the computer in terms of memory accesses. If the CRT terminal is

located remotely from the computer, communication line speeds prohibit refreshing from the computer's internal memory and force the use of a local refresh buffer memory or a storage-type CRT. CRT terminals have the advantages of speed, high reliability, quietness, and ease of formatting and editing information, but they have the serious disadvantage of not providing a permanent hard copy unless some other device is added for this purpose.

Graphic terminals are primarily of two types--CRT displays and electro-mechanical plotters. CRT terminals for graphic applications require the ability to draw lines and move spots on the face of the CRT, as well as printing alpha-numeric characters. Since most graphic applications require a two-way interaction between the user and the computer, some form of graphic input, such as a light pen, a RAND tablet, or a cursor, is required. Most graphic applications also require an alphanumeric keyboard for manual input. The deflection speed require-ments are usually much higher for graphic terminals because of the need for moving the spot from any point on the face of the tube to any other point in contrast to a strictly alphanumeric terminal in which characters are normally printed in successive positions on successive lines in an orderly manner. As a result of the higher speeds, the random positioning, the need for drawing vectors and in some cases curved lines, the larger screen sizes usually required, and the graphic input requirements, graphic CRT terminals are considerably more expensive than the simpler alphanumeric terminals. Lower priced graphic terminals sometimes use a storage tube to avoid the requirement for a buffer memory and are typically used in remote applications where the speeds are limited by the transmission capa-bilities of conventional telephone lines.

The requirements for handling and manipulating graphic information has led to the inclusion of small computers in some CRT terminals for graphic applications. These are discussed subsequently under the heading "Smart Terminals". Because of their relatively slow speed and the hard copy output, plotter terminals are used more commonly for providing a permanent graphic output of computer generated information rather than as a means of providing man-machine interaction. Plotters operated locally on-line with a computer are considered input/output equipment and will not be considered further here.

Alphanumeric CRT terminals are used in the same applications as keyboard/printer terminals. Although keyboard/printer terminals are sometimes used in reservation systems, credit reference systems, on-line banking systems, and other remote inquiry systems, CRT/keyboard terminals are better for such inquiry or interrogation-type operations. Usually, either hard copy is not required or it is very limited in such applications, and the higher speed of an alphanumeric CRT terminal is a major advantage. The lower maintenance requirements and the lower noise level of CRT terminals are also advantages in these applications.

CRT terminals are also more attractive than keyboard/printer terminals for message composition, text editing, and computer-aided instruction. The text or other material is usually stored in the central computer, and requirements for hard copy at the terminal are limited. The higher speed and greater editing and formatting flexibility offered by the CRT terminal are the major advantages for these applications.

CRT terminals are required in many interactive graphic applications. The use of graphic CRT terminals is becoming widespread in many design automation and computer-aided design applications which require very sophisticated high speed terminals. Plotters are frequently required also in these applications to provide high-quality hard copy.

Typical applications for graphic CRT terminals include design automation in: aerospace, architecture, automotive, electronics, civil engineering, structural engineering, and naval engineering; scientific problem solving in: astronomy, chemistry, geology, geophysics, medicine, meteorology, oceanography, and physics; business graphics such as: sales forecasts, financial projections, inventory management and personnel statistics.

Major advances in alphanumeric CRT terminals will be in cost reductions permitted by the use of semiconductor LSI techniques for control logic and buffer functions. Although there are several alternative technologies to CRT terminals

(e.g., plasma panel displays, light-emitting diode (LED) displays, etc.), none of these are expected to have a serious impact on the market for CRT displays during the next few years. The one exception to this is found in displays having very small numbers of characters (e.g., less than 256 characters) where plasma panel and LED techniques are being used.

Remote-Batch Terminals

Basically, remote batch terminals are assemblies of I/O and peripheral equipments at the remote user's facility which are connected via communication lines to a central computer at another location. In most systems of this type all computation and data processing is performed by the central computer with data read from input devices, such as punched cards or magnetic tape, at the remote-batch terminal, and the resulting output provided at the remote terminal by output devices, such as card punches or line printers. To the user, the remote-batch terminal appears as a complete batch-processing computer facility with I/O data processing and storage capability. In fact, however, the data processing and a major portion of the storage are provided by the central computer, while the I/O functions are provided by the remote-batch terminal. Although many people would consider a keyboard/printer terminal used in a batch-processing mode, rather than an interactive mode, to be a remote-batch terminal, the term is restricted here to those terminals which include card or tape input and printer output.

The two major approaches to remote-batch terminals are:

1. simple terminals with hard-wired control and I/O equipment; and
2. terminals which include a minicomputer for stored program control and preprocessing, with I/O equipment plus other peripherals, such as a disk file.

In the simple terminals with hardwired control, data are read from the input devices in the remote-batch terminal, transmitted to the computer over a communication line, and processed at the central computer. The results are then transmitted over the communication line back to the remote-batch terminal where the output devices produce the output records. At the remote site the control of the individual items of I/O equipment and the communication line is handled by special logic and buffers designed specifically for this purpose. For relatively simple remote-batch terminals, this hard-wired control may be less expensive, but it is significantly less flexible.

In more sophisticated remote-batch terminals a small computer is included to store programs, to control the I/O equipment and to do some preprocessing functions (e.g., code conversion and editing) and output processing functions (e.g., formatting). In such terminals the use of a minicomputer reduces the amounts of data transmitted to and from the larger central computer, provides flexible control, and provides editing and formatting for the I/O equipment.

Applications for remote-batch terminals are usually the same as those for conventional batch-processing computer and data-processor systems. Remote-batch configurations terminals do not offer any significantly new type of computing capability; rather, remote-batch terminals offer convenience for the user in providing I/O equipment in his facility while still permitting him to share a larger computer capability than he could justify in a local on-site computer.

In a sense remote-batch terminals offer the same capability to the user as a large service bureau but with the I/O equipment in his facility, thus providing very fast response and eliminating the need for taking data back and forth between the user's facility and the service bureau. Remote-batch terminals extend the facilities and capabilities of service bureaus by permitting customers to input and output their data in their own facilities while using part of the service bureau's large computer. This type of operation is frequently referred to as "remote job entry".

Advances in remote-batch terminals are tied almost directly to advances in conventional I/O equipment such as punched-card readers and punches, printers, magnetic-tape equipments, etc. The primary advance anticipated in remote-batch terminals is the increased use of small computers which will provide processing capability in the terminal as well as control and buffering functions.

Factory Data Collection Terminals

Point-of-sale terminals and factory data collection terminals are the two major types of transaction terminals or source data collection terminals in use today. Other special transaction or source data collection terminals with smaller usage are discussed later under "miscellaneous terminals", but expanded use is anticipated for some of those in future years.

Factory data collection terminals permit capturing data, such as inventory receipts and withdrawals, employee's time worked on specific jobs, machine time used on specific jobs, etc., at the source rather than recording this data manually which would require a later transcription to machine readable form for entry into the computer. This not only facilitates the operation on the factory floor, but also provides up to the minute on-line data and eliminates the keypunch or other transcription operation. The on-line capture of data permits more current reports for factory management and permits interrogation for more recent status. However, it is important to note that the potential cost advantage in using factory data collection systems is not merely that of eliminating the keypunch or transcription operations but also includes better inventory control and improved factory operation.

The major advances anticipated in factory data collection terminals within the next few years are lower costs and improved data capture techniques, such as magnetic badge readers. The terminal may later be tied directly to production machinery to permit capturing the data automatically but this is further in the future.

In a typical factory data collection system the terminal devices are interfaced to a local computer or to a special stored program process which dynamically polls the terminals and simultaneously edits, verifies, formats, augments, validates and routes the information. The terminals and processor can either operate as an off-line system or can operate on-line with the central computer or data processor. In off-line operation data is recorded from the special purpose processor on a magnetic tape which is later entered into the company's general purpose computer or data processing system. The source data entry terminals for input and output of variable data are designed to be easily usable by untrained operators, such as factory workers and supervisors. Human engineering of the terminal is particularly important for this application.

Applications for factory data collection terminals are found in electronics manufacturing companies, aircraft and aerospace companies, automotive companies, pharmaceutical companies, shipyards, and light and heavy manufacturing industries.

Miscellaneous Types of Terminals

In addition to the terminals that fall into the categories discussed previously, there are a number of other types of terminals which cannot be placed easily in these general categories. Most of these are terminals which are designed for specialized applications, which are relatively new and little used at present, or which have very small sales volumes. Examples are:

- Portable terminals
- Voice terminals
- Real-time terminals
- OCR and MICR terminals
- Meter-reading terminals
- Teller terminals for banks
- Hospital terminals

Most of these are some specialized form of data collection terminal, but real-time terminals and other special types of terminals are also included.

Specialized data collection terminals include terminals for hospitals and teller terminals for banks. A typical teller terminal prints a journal tape and a transaction ticket simultaneously, prints in the customer's passbook, provides alphanumeric and 10 key keyboards, displays 256 characters on a CRT screen for inquiries, provides function keys, and offers an optional feature that automatically reads account numbers and current balance from a magnetic tape strip in the passbook.

"Real-time" terminals are used to connect remote instrumentation and control systems into a central computer at a different location. Systems are now being developed in which the data are acquired by a remote terminal at the facility where the instruments are located and then transmitted over communication lines to a larger computer at a central location. Most of the computing, processing, and storage operations are performed at the central computer with the necessary results being sent back over communication lines to the remote terminal. The remote terminal provides the necessary control signals to the instruments and control devices and may provide some preprocessing or other local computation.

A distinction is made here between real-time terminals and transaction or source data collection terminals. Those used for data acquisition, instrumentation, and control applications where the primary interface is with sensory instruments, and control devices, are considered real-time terminals. Source data collection terminals which interface with humans instead of instruments are considered transaction terminals. These accept inputs from human oriented devices, such as keyboards or badge readers in a contrast to instruments and sensors.

Another interesting example of a special terminal, which has a large potential market, is a terminal for reading utility meters, such as gas, water, and electric, for billing purposes. These are significantly different from the real-time terminals discussed above in at least two respects--they usually read only a single meter rather than a collection of instruments and they are polled rather than normally operating on-line. They are polled infrequently (usually once a month) and may be polled at off hours, such as in the evening. Several problems must be overcome before this type of terminal is widely accepted--particularly the cost and the method of polling or collecting data from the terminal. One approach that has been proposed is to use the customer's conventional telephone line briefly in the evening hours, but the present telephone tariff structure makes this unreasonable from a cost standpoint. There is no adequate provision in the tariff structure for a device that is permanently connected to the telephone line, but which is actually used only once a month and then only for a few seconds. An alternative is to use a wireless transmitter approach in which a small low-powered transmitter is included as part of the terminal connected to the meter. In this approach a truck would tour the area once a month to poll the meter terminals. A similar approach has been proposed in which an airplane flying over the area would poll the transmitters in the terminals.

Major opportunities will exist in the terminal market for special-purpose user-oriented terminals whose features and characteristics are tailored to the needs of specific types of users or applications. There are many kinds of applications where general-purpose keyboard/printer or CRT terminals are not adequate for the users' requirements. In these cases, special terminals are required. The point-of-sale terminal is an example of this which has matured to the point that it has become a major market. Other examples are found in accounting firms, large libraries, hospitals, doctors' offices, lawyers' offices, civil engineers, architects, schools, banks, etc.

Point-of-Sale and Credit-Verification Terminals

One of the major developments in the computer field during the past two or three years has been the extent to which point-of-sale terminals and related credit verification terminals have achieved success in the market place in applications such as department store sales and super market check-out. These, and other transaction or source data collection terminals, must provide a very easy interface for an unskilled operator, must have built-in means of controlling or improving the users' accuracy, and in many cases must provide a means of automatically reading some form of coded label or tag on the merchandise. Frequently these terminals can be hard wired to a computer in the same physical facility, but in other cases a communications line tie-in to a remotely located computer is required. There will be a relatively large number of different terminal designs for source data collection, since it is particularly important that the features and characteristics of the terminal be tailored to the requirements of the application. The primary objectives in the use of such terminals

are to capture data at its source and to improve response time and accuracy. Point-of-sale and credit-verification terminals provide an excellent sample of transaction terminals and illustrate the trend toward user oriented terminals designed for specific types of applications.

"General Purpose" Point-of-Sale Terminals

The largest merchandising chain in the U.S. is installing point-of-sale terminals in most of their stores in a $750 million five year program that will include 30,000 to 40,000 electronic cash registers by 1976 controlled by more than 640 minicomputers and 33 larger computers. The system will reduce credit losses and inventory shortages in addition to facilitating check-out operations and improving the availability of products on store shelves. It will provide up to date sales data for the company's buyers and a related inventory management system for 41,800 merchandise departments in all the company's stores in the U.S. The system will also tie together over 1200 catalogue sales offices, warehouses, business offices, and 1000 of their largest suppliers. This company is also testing hand-held wands for reading magnetically encoded merchandise tickets in about 20% of their stores. Other large merchandising chains and many department store and discount store chains are installing point-of-sale terminals.

The major deficiency in the functional use of point-of-sale terminals at present is in the automatic reading of merchandise tags. Several techniques, including the conventional Kimbal punched tag, are being used at present but no standard has emerged. The two major techniques used today for automatically reading merchandise tags and credit cards are magnetic encoding and optical bar encoding. In addition, for credit verification terminals in gas station applications, the embossed data on the card is read. Serious consideration is now being given to optically reading printed characters in a standardized font with a hand-held optical wand. This will permit the use of codes which can be read by store employees and customers as well as the terminal. It will also permit merchandise tags to be prepared by standard printing techniques.

Many in the industry believe that magnetically encoded tags will eventually become the standard because of the advantages of reliability, ease of encoding, and recording density (hence capacity) offered by magnetic tags and cards. However, sufficiently low cost processes for recording the merchandise tags have yet to be developed and installed on a commercial basis. Another advantage of magnetically encoded tags is the compatibility they offer with magnetically encoded credit cords, magnetically encoded employee badges, and other data that can be magnetically encoded (e.g., department identification) and affixed to the terminal itself.

The flexibility offered by magnetic encoding is illustrated by an interesting point-of-sale terminal which may provide an indication of the direction user oriented terminals will take in the future. This point-of-sale terminal uses an all magnetic system. The two major components are a tag maker which makes tags on a strip of gummed labels. Each tag has the identification number (stock keeping unit) and the price magnetically encoded on a magnetic surface which is over-coated with a white surface on which the identification number, item name and price printed for manual reading. The tag maker prepares these labels from punched card input. The second unit in the system is the point-of-sale recorder which includes indicator lights and printed information to lead the operator through the steps necessary to complete a transaction. The entire transaction is completed by the operator by moving the hand-held magnetic wand across the appropriate magnetic strips to accomplish the data entry. The operator scans the wand across the merchandise tag to enter the stock keeping unit (SKU) number and price, across the customer's credit card to enter the customer ID number, across the employee badge to enter the employee's number, across a strip on the terminal to enter the department number, and then across a series of strips on the terminal to indicate the specific nature of the transaction (e.g., credit or cash, sale or return, taxable or non-taxable, etc.). All of these steps are encoded on magnetic strips on the terminal. At the end of the transaction the terminal extends the amounts, calculates the tax, performs other necessary arithmetic

operations, and prints the sales slip. The terminal has a small 10 key keyboard for special cases and exceptions, but this is used so infrequently that it is placed under a cover. The terminal can operate in either a stand-alone mode or an on-line mode with a minicomputer system. When operating on-line with the minicomputer system, the terminal can also make credit checks and record inventory data via the minicomputer. This all magnetic approach avoids requiring department store clerks to key input data.

A more typical point-of-sale terminal announced recently provides the following features:

- Selective itemization
- Tax table look up
- Positive, negative, or positive/negative credit verification
- Calculations of decimal quantities
- Minimum/maximum digit entry
- Multiple check digit calculations
- Transaction counter
- International currency calculations.

The dual printer in this terminal permits simultaneous print out of an audit tape and a multiple cash receipt or sales check that can be loaded from the side. The printer mechanism uses a 9 x 7 dot matrix and operates at a 120 characters per second. It prints 10 characters per inch, each .07 x .10 inches. A programmable-read-only memory (PROM) controls the printer in printing a 29 character line upside down and backward for sales receipts and right side up for the audit tape.

Typical functions that must be performed in point-of-sale terminals include the following:

- Credit verification
- Capture of billing data
- Capture of inventory management data
- Special calculations -- e.g., tax computation, employee discounts, etc.
- Conventional cash register functions such as:

Change computation	Amount entry
Department totals	Produce receipts
Produce audit trail	Activity counters
Sales totals (itemization)	Charge, non-add
Display of amounts and totals	Refunds and credit

A number of point-of-sale terminals include credit verification capability when connected on-line to a computer system and credit data base. If the point-of-sale terminal does not include the credit verification function, it can operate off-line and record data on a magnetic tape or cassette. The terminal can then be polled by a central computer over a communications line at night or the magnetic tape can be sent to the central computer. In some cases, an off-line terminal prints a journal tape which is later read using OCR techniques. In any event, off-line capability is required to permit continuous sales activity when the central computer is inoperative.

Credit Verification Terminals

Ultimately, in department store applications the credit verification function will be completely absorbed by the point-of-sale terminal and the separate simple credit verification terminal used in some department stores at present will disappear. However, requirements for relatively simple credit verification terminals will continue to two major applications--oil company credit cards for gasoline stations and bank credit cards for small merchants and other stores that accept bank credit cards. In this context, "travel and entertainment" credit cards are included with bank credit cards. Initially, most credit verification terminals will only perform the credit verification function, but ultimately most of these terminals will also capture billing data for descriptive billing. Typical terminal functions required for credit verification and capture of billing data include the following:

- Embossed or magnetic card reader
- Manual numeric input of sales amount

Manual or automatic numeric input of card company code
Present digits for merchant identification
Manual input of card number
Imprint sales invoice
Display of manual input data
Display of computer response
Display and imprint credit verification number
Variable function keys
Identification control character for terminal type
Buffer data prior to transmit
Standard communication format (ASCII/300 baud)
Communications and polling for dedicated line
Optional auto dial communications.

The major distinguishing factor between different credit verification terminals at present is the technology used for automatically reading the identification number from the credit card. The card number input technologies used in credit verification terminals at present include:

Magnetic card reader
Embossed card reader
Optical card reader
Punched card reader
Manual card number input

Special Purpose Point-of-Sale Terminals

In addition to the more or less general purpose point-of-sale and credit-verification terminals discussed previously, two specialized point-of-sale terminal markets are becoming important--supermarket check-out systems and fast food (McDonalds, etc.) or restaurant chains. At least six or seven companies in the U.S. have either developed or are working on a specialized terminal for check out stands in super markets and other large grocery stores.

One of the major factors holding back the widespread use of electronic terminals in super markets has been the lack of agreement on a standard coding scheme for identifying the item number (universal produce code or UPC). Such a standard is being developed by a trade association of the grocery industry (the Supermarket Institute) but has not been finalized. In addition to the standardization of a technique for recording the code on the merchandise and scanning it electronically. Techniques under consideration include optical bar codes, circular codes, and radial codes. The circular and radial codes have the advantage of being omni-directional and can be scanned in any direction. A standard code will permit the supplier to print the code on the merchandise, thus eliminating marking in the store. Reading this code in the terminal can permit the price to be looked up by the computer providing easy handling of price changes and special sales price. Large scale tests are underway in several stores using different systems and techniques.

The second specialized point-of-sale terminal market that will be very large is that for restaurant chains, particularly those called "fast food" chains, such as McDonalds, Jack-in-the-Box, Kentucky Fried Chicken, etc., which have the common characteristic of a very limited fixed menu which simplifies the identification of the food items at the order taker or cashier's stand. No unit has received wide acceptance in this application to date but several companies have developed systems. The three major functions required in terminals for the fast food industry are the totalling of the sales amount for the customer, keeping continuous account of total sales at each point of sale, and keeping records of sales of each item for inventory purposes so that each store can receive supplies the next morning to replace those used the previous day.

Smart Terminals

The term 'smart terminal' is used here to identify an interactive terminal in which part of the processing is accomplished by a small computer or processor contained in the terminal itself. This type of terminal is sometimes referred to

in the literature as an "intelligent terminal". To be considered a smart terminal the computing capability of the minicomputer in the terminal must be available to the user in a way that permits him to program it to perform part of his unique application. Terminals that include a programmed processor only to control terminal functions and provide required logic and buffer capability (e.g., refresh storage for a CRT, character generation, vector generation, etc.) are not considered smart terminals in the context of this discussion. Mere substitution of a stored program processor for hard-wired control logic and buffer storage in a terminal does not qualify it as a smart terminal; rather, part of the internal computer capability must be available to process subroutines or portions of programs written by the user.

An interactive smart terminal has the following characteristics:

1. Self-contained storage
2. User interaction--with the terminal or the central computer
3. Stored program
4. Part of processing accomplished in the terminal
5. On-line via communications line with large central computer and data base
6. Human oriented input--keyboard, light pen, etc.
7. Human-oriented output--serial printer, CRT, etc.

Although the term "smart terminal" is restricted in this discussion to interactive terminals, a remote-batch terminal could employ the same approach--using a small computer whose capability is available to the user in addition to serving as a control and buffer device. Smart terminals are more of a concept than a unique functional terminal category. In a few years it will be difficult to consider smart terminals as a separate terminal category, since the use of internal processing capability will pervade most types of terminals.

Rationale for Smart Terminals

Rapid changes in semiconductor technology have significant impact on the relative costs of different parts of a computer and communications system. While the logic and internal storage costs have decreased very rapidly, mass storage costs have decreased less rapidly and communication costs have remained essentially constant; but in addition, software has become an ever increasing percentage of the total cost of the computer system. There is no longer any severe economic penalty in distributing processing capability to individual terminals.

In time-sharing systems and other multi-access systems, there are a number of sound functional reasons for centralizing the data base--particularly in those cases where several users may alter the data base as well as accessing it. However, there are no strong functional reasons for centralizing the computing and processing functions themselves. The decision as to whether to centralize the individual user's computation and processing operation is an economic one. With present trends in digital equipment costs on the one hand versus communication costs on the other hand, it will be economically attractive to do as much of the computing and processing locally as possible in order to minimize the data transmission requirements and hence the communication costs.

The program currently being executed (and possibly other frequently used programs) is stored locally at the terminal along with part of the users data base. The central computer facility provides the following:

1. Mass storage -- to handle shared data bases and to achieve the economy of scale that still applies to mass storage devices.
2. Expensive and/or infrequently used peripheral equipments -- to provide high-speed line printers, high-speed plotters, computer output-to-microfilm, etc. that are not economically feasible at the individual terminal.
3. Computing and processing capability -- to handle data base manipulation and very large problems or parts of problems that are too large to be handled in the terminal.

Graphic Smart Terminals

A typical smart terminal for graphic applications may include a small computer with several thousand words of internal storage, a CRT display, a keyboard,

a graphic input device, a printer, and a communication modem. It may also include a small disk or some type of bulk storage.

Applications for graphic smart terminals include those requiring interactive design in design automation and interactive problem solving and modeling in scientific computation.

Smart terminal advantages in design automation are perhaps greater in the automatic drafting phase than in the actual design process itself because of the ease with which changes can be made that affect a large number of related drawings.

Small Business Machines and Smart Terminals

A number of companies are working on sophisticated business oriented smart terminals and small dedicated information systems based on minicomputers. One of the more advanced examples of alphanumeric smart terminals announced recently is a relatively small, but very sophisticated intelligent terminal which can also operate as a stand-alone small business system. It includes the following equipments:

- Split platen character serial printer
- 14,336 to 65,536 bytes of MOS semiconductor memory
- Communications interface
- Options
  - Magnetic tape cassetts (up to 4)
  - Dual data communications operations permitting simultaneous and independent data transmissions
- Compatible magnetic tape unit
- 90/180 line per minute line printer
- Punched paper tape
- 96 column card reader and punch
- 80 column card reader and punch
- 256 character plasma panel display
- Magnetic ledger card handler and reader.

More and more companies are beginning to use smart terminals of this type as terminals at night but as stand-alone systems during the day for functions (e.g., inventory control, order processing, etc.), where the local data base is maintained on site. The terminal can then be polled at night (over a communications line) by a large central computer to collect the data captured by the terminal, which acted as a stand-alone system during the day, and to update the local data base at the terminal. This has a number of advantages over the use of a simpler terminal operating continuously on line, including:

- Significant lower communication costs
- Lower error rate
- Maintaining local data bases and files
- Maintaining local operations and processing despite communication system or central processor maintenance or down time
- Data validation and correction at the source prior to transmission.

The use of small local systems in this manner also significantly reduces the computation and processing load on the large central computer and permits software changes to be made more easily to accommodate any unique local conditions.

Hence, in considering smart or intelligent terminals for commercial, financial and industrial applications, it is important that a manufacturer address both the small business machine and the smart terminal market because of the close relationship between them.

With current trends in technology, it is reasonable to expect a small smart terminal in the future to include:

- a 4000 to 65,000 byte minicomputer
- a 300 to 122 bit/s modem and communication interface
- a keyboard and function keys
- a character serial printer (30-60 cps)
- one or two magnetic tape cassettes
- an optional CRT (1000 to 2000 characters)
- an optional light pen.

These technology trends can also reasonably be expected to provide a larger smart terminal including:

a 32,000 to 128,000 byte minicomputer
a 9600 bit/s modem
a keyboard and function keys
a character serial printer or other hard device
magnetic tape cassettes
a card reader (100-300 cpm)
an optional graphic CRT display and light pen
optional disc file
optional compatible magnetic tape units
an optional line printer (100-300 lpm).

5. CONCLUSIONS

Advances in hardware technology, particularly LSI semiconductor technology, will significantly change the overall design of computer systems, increase the utilization of computers by small users, decentralize computer facilities, and lower the cost of computing and processing capability. Input/output hardware will improve in performance and decline in cost but in an evolutionary manner. LSI technology will afford improvements in the performance and cost of input/output equipment but the impact will be felt primarily as reduced requirements for input/output equipment relative to other portions of the system.

The major impact of technology advances will be felt in the areas of memories and terminals. LSI memory technology will offer significantly lower costs and higher speeds for main internal memories. LSI logic and buffers, coupled with advances in magnetic surface recording, will permit increasing storage density in magnetic disc files with a consequent increase in capacity and decrease in cost per bit. LSI technology is already reducing the cost of terminals and permitting the inclusion of extensive logic and storage to provide "smart" or "intelligent" terminals with internal stored program processing capability.

During the next few years the user can anticipate several significant system trends including:

1. Increased availability and utilization of on-line communications oriented systems.
2. Increased use of systems with larger and larger on-line data bases to permit computer access and interrogation of massive files and to reduce input/output requirements.
3. Increased use of on-line data entry, source data collection, and special-purpose applications oriented terminals.
4. Increased use of smart or intelligent terminals with extensive processing and storage capability, including terminals that can operate in a stand-alone mode as well as providing terminal capability when required.

Remote terminal and mass memory hardware developments, plus parallel developments in on-line processing, man-machine interaction, multi-user systems, communications oriented systems and distributed processing, will make the capabilities of computers more and more available to the laymen. These developments, fueled by cost reductions resulting from advances in LSI semiconductor technology are making computers and terminals more pervasive in our society. They are leading to ever expanding applications of computers, increased exposure by the laymen to computer interfaces, and increased utilization and dependence upon computer technology by society. The success of these trends depends upon the extent to which remote terminals provide easy to use interfaces for these casual users and upon the extent to which software development makes the computer system transparent to the casual user rather than standing between him and a solution to his problem.

DISCUSSION

STEEL:

You have talked of 5 years in the future; can you give an indication of what 15 years in the future might look like?

HOBBS:

The further we get out the harder it is to see the trend. We tend to overestimate what we can do in the near future from a technology standpoint and we underestimate what we can do in the long range. I would expect that you would find very very significant development if you look into the 1980's. The developments will be primarily ones of reduced cost and the availability of computers to the casual user and the layman. Industry has to make it possible for someone who has no interest in computers easily to interface with the computer. I would expect the performance to approach a plateau during the period. I don't believe the same would be true of cost. Referring back to the semi-conductor memory, the state of the art now is a thousand bits on a single semi-conductor package. By next year we expect that to be four thousand bits. By 1978 I would expect that to be eight thousand, perhaps 16 thousand bits and past that point perhaps a plateau in the number of bits we can put on a single chip. By using techniques to combine several chips within one package, we could expect the capacity of the package to continue to increase. With respect to mass memory, the disk memory, looking more than 5 years ahead we are again approaching a plateau in increasing the bit density of a storage media. Improvements in head positioning technology will permit a much higher track density and therefore a larger storage capacity in the file. In the time phase you are talking about we probably will be looking at some kind of optical memory.

LOWENTHAL:

You have given us all hope as far as cost per bit and so on, but you said that the average access time will probably not improve much over the next 5 years. If I have a very large data base in 1976 which has to be accessed much faster than current moving head disks, what alternatives will I have?

HOBBS:

Your alternative is to access in parallel a number of different devices. Your only solution for faster access is to pay the price for a head-per-track device.

MOEHRKE:

I get very concerned with the problem of distributing the logic between the main computer and the intelligent terminal. As the user sees it, it's one application. The users have a great trend for expecting more and more, once they see what they can get. The problem is dividing this logic between the machines. I see real difficulty in continually sliding things back and forth over this interface.

HOBBS:

I put the burden back on you as programmers when I said that you must make the systems transparent to the user. The user must not be concerned whether a particular part of his task is being done in a terminal or in the central processor. Eventually, in the future, we will have compiler systems that distribute the work between the central processor and the terminal. We are a long way from there at the moment.

GOSDEN:

Those maligned users had that same problem before EDP came along. They have been able to divide work between different people in different locations in different parts of the country in different parts of a business. I do not really think it is a very big problem from a systems point of view. I think the real problem is to reject the syndrome that a big system has to grow like a cancer and end up being what we call a kludge. I believe there are many advantages to splitting up systems into subsystems that are physically separate.

HOBBS:

The use of a very intelligent terminal permits you to compartmentalize the overall software in a way that I think would not be possible if you had a single large processor.

Data Base Management Systems, D. A. Jardine (ed.).

MANAGEMENT AND ECONOMICS OF DATA BASE MANAGEMENT SYSTEMS*

J. C. EMERY and H. L. MORGAN
The Wharton School
University of Pennsylvania
Philadelphia, Pennsylvania.

INTRODUCTION

It is becoming increasingly clear that data base management systems (DBMS) will play a central role in information systems of the future. They have already had a significant impact on some of the more advanced current systems; in not too many years they will be almost universally accepted as standard system software.

The use of a DBMS can require a major commitment of organizational effort and resources. This is typically not the case when a self-contained DBMS is used to process ad hoc inquiries; although such a system can be enormously useful, it does not usually have a major impact on the organization or its information system. On the other hand, if a host language DBMS is used as the central core around which the organization's information system is built, then the impact can be profound.

The issue of which DBMS should be selected and how it should be used is too important to be left solely in the hands of the technical staff; management and users must also play a part. This paper will focus on some of the important managerial and economic issues that should be considered. We first examine the benefits and costs of a DBMS, and then we discuss its managerial implications.

BENEFITS FROM A DBMS

The implementation of a DBMS should be viewed as a major investment yielding certain benefits and entailing certain costs. The benefits and costs should be spelled out clearly and (insofar as possible) expressed in monetary terms. The objective should be to choose that DBMS that provides the most attractive cost/benefit performance (Emery (1971).

Reducation in application programming effort. The major cost saving gained by the use of DBMS is the reduction in the manpower required to implement debugged application programs. One large firm has estimated that it takes about 20% as much effort to program and debug an ad hoc inquiry in a self-contained DBMS language as it does in COBOL. Application programs written in conventional programming languages, but which access the data base through a host-language DBMS, generally are considerably cheaper and faster to implement than programs that have to handle their own data management functions.

Several factors contribute to the greater productivity of a programmer using a DBMS. First, the details of input/output are no longer part of the applications program. When using a host language system, the interface to the DBMS is typically a CALLed subprogram. Since much of file processing is by its very nature I/O oriented, the ability to avoid the details of I/O provides a major savings.

Second, the application programmer no longer need concern himself with the details of file organization, data descriptions, etc. The data division often accounts for a major share of a COBOL program, and so the elimination of this portion can substantially reduce programming effort. Naturally, the data definition must still be handled, but for the data base as a whole rather than on an application-by-application basis.

* The work reported here was supported in part by the Office of Naval Research; project NR 049-272.

Finally, many of the functions that an application programmer might normally have to provide have already been included as standard functions in the DBMS. Security, for example, can be handled entirely at the data base level through the DBMS. The powerful language features provided by self-contained systems such as MARK-IV and ASAP can dramatically reduce report preparation time over that required using standard programming techniques.

Permits a common data base. A less easily quantified--but ultimately perhaps the most important--set of benefits comes from the ability to implement a common data base using a DBMS. Without such a system it is virtually impossible to maintain the degree of control over application programs and files necessary to establish a common data base. In the absence or the ability to manage a common data base, communication among subsystems is difficult and expensive. This inhibits integration, results in duplication of data entry and storage, and raises serious problems of reconciliation among fragmented subsystems.

While a "common" data base generally does not lead to a complete avoidance of duplicate storage of data, sharp reductions are likely in the total volume of storage. Consider typical university records, in which the name and address of a student often appear in half a dozen or more files. If each occurrence takes 100 characters, and there are 20,000 students, 12 million bytes are used to store names and addresses. With a common data base, and using three characters for a pointer to a single copy of the name and address information, a savings of 9.7 million bytes can be made. In addition, updating costs are reduced, since only one record must be changed when the name or address is modified (when a student moves to a new location, for example). Substantial intangible benefits are also reaped by having each of the subsystems refer to the same person by the same name. This avoids such problems, for example, of having the Registrar refer to a student as John J. Smith while the Controller calls him J. J. Smith, and thus possibly facing major difficulties in recognizing that the two names refer to the same person.

The development of a common data base implies that data base definitions, the logical and physical structure of the data base, backup, security, etc., are under some form of central control. Rather than spreading money and effort across multiple fragmented files, attention can be focused on the single common data base. For example, designers can concentrate on a very efficient physical storage representation for a heavily used single file, rather than spending the same (or more) effort on several less efficient designs of independent files. Monitoring techniques that might not be justifiable on the basis of any single application may prove feasible with a common data base. The benefits that then accrue are effectively multiplied by the number of different subsystems that use the data base.

The cost of backing up many small files, in both time and operational terms, can be substantial. With a common data base, a single backup procedure can often suffice for all applications, again yielding a multiplicative benefit.

Greater flexibility, maintainability, portability, and reliability. To require individual applications to provide a high degree of flexibility and data independence would be costly and probably unjustified. However, such flexibility is inherent in a well designed DBMS. It is difficult to estimate in dollar terms the value of being able to respond to changes in information needs or advances in information technology, but it is very substantial.

A software firm that markets a DBMS has a strong incentive to maintain the system, make it portable across generations and vendors, and insure its reliability. Although all DBMS have weaknesses in these respects, they are apt to provide better performance than systems developed and maintained in-house. Again, these benefits are difficult or impossible to quantify, but they can be significant nonetheless.

COSTS OF A DBMS

Acquiring and installing the DBMS.. Not the least of the costs associated

with DBMS are those incurred in the process of selecting and acquiring the DBMS. A rational selection procedure involves the ranking of various alternative systems on the basis of a set of requirements. In one case in which a detailed analysis was performed, approximately three man-months were spent in evaluating six alternative systems. After a preliminary selection, the top candidates were benchmarked. The preparation of a good set of benchmarks involved an additional man-month of work. The organization thus had a significant investment in the DBMS even before it came in the door.

The cost of leasing or purchasing a DBMS varies widely. The ASAP system can be purchased for $12,000 (and DYL-250 costs only a dollar a day), while ADABAS costs well over $100,000. Byrnes and Steig give a survey of various systems from the standpoint of cost and overall features (Byrnes (1972)).

Once the system has been acquired, there are major costs of getting the system to the point at which it can be used within the organization. First comes the training of the technical staff in the use and maintenance of the new package. Then the system must be installed. Some systems require only a simple link edit, while others require a full system generation, with all of its attendant computer time and personnel expense. The creation and distribution of user-level documentation, which relates the DBMS to the organization's own data base, can add considerably to costs.

If a large amount of file conversion must be performed, costs can mount rapidly. Either a utility package for converting files needs to be acquired, or programs must be written and debugged to perform the conversions. Often it is less expensive to create a file from scratch; in this case substantial data collection costs can be incurred. In addition, precautions need to be established for maintaining the integrity and backup of data during the early phases of running with the DBMS.

If the organization is going to have an effective common data base, a "Data Base Administrator" must be appointed. The costs of maintaining such an office include specifying data definitions, documenting the data base, establishing and running backup procedures, monitoring data entry and interface procedures, and specifying and enforcing standards.

Operating costs. Once the DBMS and application programs have been installed, operating costs begin to appear. Some of the DBMS require large amounts of core. IMS-2 for example, typically requires 250K bytes of dedicated core (and some of the full-blown versions require much more than this). MARK-IV, on the other hand, can operate in a 32K DOS system (but provides, of course, more limited capabilities).

The CPU overhead associated with a DBMS also needs to be considered. In some cases the use of a host language system adds overhead by increasing the number of accesses to auxiliary storage. On the other hand, some systems are more efficient than the alternative they replace (e.g., TOTAL replacing ISAM/OS), so it is difficult to generalize about efficiency.

Common data bases often imply many linkages among data items. This, in turn, usually implies random access storage. There may also be additional requirements for peripheral devices due to the DBMS. All of these requirements add to costs.

Finally, there are the continuing operating costs of data base administration. The administrator should collect statistics on the usage being made of the data base, log transactions for backup and recovery purposes, perform security checks, and maintain a general discipline over the data base.

Intangible personnel costs. In addition to the tangible costs of a DBMS, there can also be important intangible costs as well. Many technical staffs have a strong resistance to change. The introduction of a DBMS causes not only change, in the sense of a new language, but also imposes many restrictions. Particularly the "bit twiddlers" and those who delight in optimizing I/O may feel a strong sense of having their wings clipped.

Some programmers refuse to use a query language subsystem, claiming that they can write it faster in COBOL or some other procedural language. At least part of their resustance is due, no doubt, to a concern that the new language does away with the need for their hard-won skills.

MANAGERIAL IMPLICATIONS

The impact that a DBMS has on an organization depends on the nature of the system and how it is used. A self-contained DBMS that only supplements the existing information system might have relatively minor managerial implications. But when the DBMS becomes the core of an integrated system it can have a significant impact on project management and the assignment of organizational responsibilities. The trend is clearly toward the use of the DBMS as an integral part of the overall system, and so management shouls be prepared to deal with the consequences.

Data Base Administrator. One of the primary motivations for adopting a DBMS is that it makes it feasible to move toward a higher degree of integration by means of a common data base shared among many users and programs. By its very nature, a common data base requires centralized control of data definitions, coding, security, and physical and logical organization.

The responsibility for these matters should rest in a single manager. Increasingly such a manager carries the title of "Data Base Administrator" or (DBA). Because of the importance and centralized nature of this function, the DBA should probably report directly to the senior executive in charge of the overall information system activities.

Giving authority for data base administration to a single person raises some fundamental organizational issues. It certainly impacts analysts and programmers, because they must learn to live within a much more tightly disciplined environment than they face with a fragmented data base. The organizational problem is particularly difficult when applications are developed on a decentraliz d basis (by management function, such as marketing, manufacturing, and accounting, for example). There is nothing incongruous about imposing a disciplined data base on decentralized application programmers--after all, programmers currently must abide by all sorts of standards and constraints--but the organization should at least understand the managerial issues when moving toward centralized data base administration. Unless sufficient backing is given to the DBA, he is bound to fail.

A good DBA should meet a number of demanding qualifications. He (or she) should certainly have technical skills, because many of the issues he must deal with are largely technical in nature. Decisions regarding data base organization for example, depend heavily on technical considerations. The DBA must also have strong managerial skills, because the decisions he makes can have profound managerial consequences. He should understand the needs of users and be skilled at negotiating with them in order to arrive at a design that provides the best compromise among conflicting requirements.

Effects on centralization versus decentralization. The centralization of data base administration is only one of the organizational consequences of adopting a DBMS. Certain other functions are also likely to be centralized. This will probably be the case with high-volume transaction processing subsystems. The DBMS, when coupled with a powerful central processor and programming language, permits some consolidation of application programs. A common data base also eases some of the problems of implmenting applications that cross existing organizational boundaries (such as an order entry subsystem that crosses the boundaries of the marketing, manufacturing, and accounting department). Furthermore, the implementation of these applications calls for a high degree of specialized skill that can often best be supplied on a centralized basis. All of these effects tend to favor greater centralization of the implementation process.

The DBMS might also favor some centralization of organizational planning and control, since the more powerful technology permits an expansion of the scope, time horizon, and detail of centralized decision making. Decentralized decision making is often justified by the limited ability of a centralized staff to obtain the necessary data and processing resources to support global planning. A powerful DBMS, integrated with large-scale decision models, overcomes some of the technical limitations that have inhibited centralization.

In some respects the DBMS can lead to the decentralization of certain functions. As query languages become more user oriented and easy to learn, many tasks conventionally perform by a centralized technical staff can be handled by the user himself (or his staff assistant). When a user can (easily) write his own program (within the constraints imposed by the query language and DBA), the system achieves the ultimate in user responsiveness.

This philosophy can be extended to decision making aids for decentralized managers. The decision makers can be provided a means of developing their own decison models that obtain the necessary input data through the DBMS. The security provisions of the DBMS can allow access to authorized data elements while denying access to data that the decision maker does not need to support his decisions. Since decision models are often idiosyncratic, decentralized development allows the model to be tailored to the peculiarities and special needs of individual decision makers.

The centralized technical staff can play an important role in achieving decentralized use of the DBMS. They can do much to make the technical complexity of the DBMS transparent to users. They have the responsibility for establishing the data base, creating the necessary user interfaces (with self-prompting data entry, for example), preparing appropriate documentation and manuals, training users, and providing continuing user support. The technical staff can partially tailor the system to a particular user by permitting him to define synonyms and compact names for standard aggregations of data elements. In short, the technical staff should provide the user with a powerful means for expressing his own information requirements without having to worry about detailed and irrelevant technical matters.

Charging for information services. The implementation of a common data base under a DBMS raises some very difficult costing problems. Such a system tends to be expensive to implement. Once installed, however, it serves a wide range of applications and the incremental cost of each new application tends to be relatively low. It thus offers a high fixed cost and low variable cost, and it provides joint benefits across many applications. Charging for the system therefore involves the allocation of fixed costs among time periods and multiple applications--the two costing problems that have traditionally plagued cost accountants.

Conventional treatment of costs can lead to the misrepresentation of costs and the motivation of inefficiencies. For example, most organizations treat the cost of implementing the DBMS as a current expense, both in measuring managerial performance as well as for tax purposes. This distorts the cost picture, since the DBMS provides future benefits. If management places too much emphasis on short-term performance, a manager may be motivated to avoid the high short-term cost of installing the DBMS.

There is no completely satisfactory way of charging for a DBMS. Nevertheless, certain principles appear to be attractive and should be more widely adopted:

- Capitalize the large fixed cost of implementing a DBMS (for tax purposes the cost can still be handled as a current expense). Depreciation can be calculated on the basis of time or use.
- Do not arbitrarily allocate implementation costs among lower-level users. Thus, if the DBMS serves the entire organization, its installation cost should be borne at the top level (to be treated as any other general overhead item).
- Operating costs should be charged for on the basis of the resources used. This requires, of course, a means of measuring resource utilization. The operating costs of using the DBMS should be treated in exactly the same way as any other program, and so the DBMS does not introduce any new problems of charging for computer operating costs.
- The cost of collecting and storing common data should be treated as corporate-wide costs if the data are widely, and fairly evenly, used throughout the organization. If this is not the case, actual use can be measured on a sample basis and costs allocated proportionately. In some cases it may be practical to charge on the basis of the number of accesses to the

data base or the volume of data transferred, but in most cases any improvement in resource allocation brought about by this refinement is probably not great enough to justify its additional cost.

CONCLUSIONS

Any extended statement at this point about the economic and managerial implications of DBMS would be repetitious and trite. Nevertheless, our major conclusions should at least be listed:

- .DBMS will be almost universally adopted within a few years.
- .Organizations will increasingly design systems with a DBMS as the central core.
- .The sharing of data will become increasingly common as DBMS mature and are more widely used.
- .A system built around a common data base will bring about profound organizational and economic changes.
- .The benefits of installing a DBMS are often difficult or impossible to estimate in monetary terms, while the costs are largely tangible.
- .The installation of a DBMS usually must be based heavily on subjective evaluation of intangible benefits.
- .It is not too early to start planning for the implementation of a powerful DBMS.

DISCUSSION

STEEL:

Some of the problems that you discussed with respect to costing go away if you make a very careful distinction between costing and charging. The hardware manufacturers clearly note the difference. How you actually allocate the real costs is one thing; how you charge is what really motivates behaviour. You may well undercharge in some situations and overcharge in others to influence the general corporate behaviour. If you make that distinction, some of these problems are substantially clarified.

MORGAN:

I agree completely with you that we have to view the charges leveled against users as a rationing device. That may mean that those charges don't bear any well defined relationship to the cost accountant's costs. A trivial example would be charging higher rates for prime time in a time sharing system. It is very difficult to justify that in a strictly algorithmic way, but as a way of rationing a scarce resource that everybody covets, we want to charge higher rates for day-time use than for night-time use.

MAYNARD:

You talked about the impact on the organization, mostly about the organization of the system, the information processing, the computer locations, the systems groups, EDP area and so on. There is a great potential for the business itself in the flexibility it has or will have, in the way it organizes its business, and how it allocates its responsibility. What thoughts do you have on decentralization or flexibility of organization of the business itself?

EMERY:

In response to an earlier comment, one of the problems in the costing area is that many cost accountants don't understand the difference between cost and price.

The government is particularly guilty of that. There are really serious problems in a university, for example, caused by the fact that the government requires charging for computer services in a very cost accounting way. The concept of pricing as a resource rationing tool is foreign to most of the cost accountants.

The impact on the organization is by no means limited to the EDP activities. In particular, data base management systems will affect the planning and control process, which I really take as the whole decision-making process in an organization. That's going to be impacted by this technology. If we have a technology that permits relatively global data bases, relatively global models, like a large LP model for example, we have the technical means to deal with more global decision making. The petroleum industry has been perhaps the prime example of where the globalness of planning has been increasing. Other industries have showed that same trend. Where we're dealing with issues that have strong interacting effects across organizational units such as production, transportation, refining, and marketing, we have a very strong incentive toward doing more global planning. The technology now permits us to have a much higher degree of centralization of planning than we have had in the past. Therefore we are going to see more of it because the cost-benefits favour it. That obviously is going to have organizational impact. It's not necessarily true that the impact will be in terms of the formal structure, but the assignment of decision will certainly be impacted.

PLAGMAN:

Earlier in your talk you mentioned personnel-oriented types of costs with respect to DBMS implementation or utilization. I would like to ask you what percentage of the total cost of the DBMS is personnel-oriented, as opposed to hardware/software costs.

MORGAN:

It would be very difficult to estimate. The one major experience I have had was not very representative because it was involved with a new project. There was a very large increase in the size of the staff developing this new system, only part of which was because of the use of a generalized data base management system. Part of the cost was getting them to fit within a constraint of this new approach: the generalized data base management system and the data base administrator. I couldn't being to say what percentage this was of the total cost except that it was quite significant. I certainly couldn't begin to allocate them.

SIBLEY:

I'm not sure how intellectually dishonest we're really being when we talk about cost-benefit analysis. In my consulting experience 90% of the companies have no idea of any sort of benefit whatsoever. Do you think really that we are moving towards reasonable cost-benefit analysis?

EMERY:

What I've tried to point out is that we should not shy away from using subjective value judgments in trying to assess the intangible benefits. Your figure 90%, I would not be prepared to agree with. I think that a higher percentage of the benefits can be expressed in monetary terms. A higher percentage still can be expressed in quantitive if not monetary terms. But there is going to be a residual amount which we can't express in montary terms.

SIBLEY:

I meant, not <u>can</u> but <u>will</u>. I'm saying, in my experience, in fact, people do

not do a benefit analysis, and are not willing to expend the amount of effort involved in order to make one.

EMERY:

What we usually get is not a cost-benefit analysis but justification for decisions that have been made previously. The problem is not so much the DP people's fault but the fault of the managers above them. Their managers are the ones who have to start fostering rational cost-benefit analysis.

MORGAN:

The fact that we are not in most organizations using good techniques of cost-benefit analysis to the extent that it can in fact be done, doesn't really say that it's really not intellectually respectable to talk about it. We should be doing these things. It's cost-beneficial to do cost-benefit studies (up to a point) but we have not been doing it.

KIRSHENBAUM:

I think it is about time we moved at least one level about charging for I/O operations and start charging users in terms of things they understand. A user cannot budget in terms of I/O operations. The people behind the scenes have to figure out the unit cost per item produced. The user or the purchaser in this case never sees those unit costs, and I don't think he should have to. Our users have been budgeting for premiums collected for about 112 years and they should be reasonably competent at doing that. They shouldn't have to think in terms of the units that the people in the back rooms use to generate the product. In the process of doing this, it is possible to move the user, and the people who are building the system to structure the systems in such a way that, while optimizing in terms of the charging algorithm, they are optimizing for the whole company.

EMERY:

That is a lot harder to do than to talk about. It is obviously desirable and I absolutely agree, if you can charge in terms of the functional service provided by the system you're much better off and it is much more meaningful to the user.

WIEDERHOLD:

When we move from costing to pricing we have to be very careful. There will be different kinds of people that will set the prices. I've seen pricing set by computer organizations that moved the organization in a direction which I felt was unfavorable to the organization. Costing is a bit safer if you don't get management involved.

EMERY:

One of the virtues of cost accountants is they go through an algorithm and they come out with a cost. It is not always the same cost and there is certainly disagreement among different cost accountants, but it has the ring of plausibility. When we start dealing with prices then it is a tougher management job. I do not personally like to see EDP groups set up as profit centers. It seems to me in a well run organization there is going to be a deficit in the EDP group because of things like seed money and developing common systems. It may not show up explicitly as a deficit because you could formally purchase services from the EDP group. I think it can be very disfunctional if there is an organizational need to allocate all cost so that the EDP group comes out with either a zero

balance or a profit.

TURNER:

How do you feel that implementing applications using a data base management system affects the risk of implementation?

EMERY:

The professional competence that is required goes up when we develop any sophisticated technology. One company I have worked with has a five year project. They're now just in the phase of bringing the first operational part of the system into existence. They're on schedule and on budget and there is every indication that they will continue to be on schedule and on budget. They have done a superb job of management. They put a lot of money and a lot of effort in doing it. One can manage technical risk in this area, as one can in Apollo or any other large system, but it does call for very high degree of management skill. If you are going to go this route, it seems to me that you have to be willing to make those investments. Otherwise, you may be better off not to try it.

MORGAN:

The large problems in getting big systems up have been in the areas of concurrency, control, input/output, and file handling. Data base systems have solved a large part of that and removed it from the applications programmer's domain. The risk of major failure has got to go down. The risks of other types of failures may increase somewhat, but the risk of not being able to do anything is way down.

Data Base Management Systems, D. A. Jardine (ed.).

PRINCIPLES OF DATA INDEPENDENCE

D. A. JARDINE
Queen's University
Kingston, Ontario

Data independence is concerned with the problems of separating application programs from some aspects of the storage and structure of data in the data base.

The motivation for this separation is the protection of investment in data and programs in a changing business and computing environment. Thus, 'how much independence' is an economic question, involving trade-offs between flexibility and efficiency.

There are two characteristics of data management systems that bear on data independence.

1. There is migration of knowledge of data relations and stored representations from the application program to systems programs.
2. Symbol resolution, ie. mapping of 'name' known to the application program to the 'name' known to system, is diffused.

I have given the name symbol resolution to what we have loosely called binding.

At this point it should be understood that we are discussing data, not reality. We must be very careful to distinguish between data relations as models of the real world and data relations in a data base and in the application programs that access it. There may not be any relation whatsoever between the data relations explicit or implicit in the combination of application program and data management system and the real world. The relations in the data base exist only because somebody stated them that way.

1 DATA RELATIONS

There are many views of data. It is the thesis of this paper that at least the following five views are necessary to an understanding of data independence and to an understanding about how we interpret data in the machine as related to data in the real world. There are:

1. Stored data relations
2. System-known logical relations
3. Program defined logical relations
4. Algorithm defined logical relations
5. End-user perceived logical relations.

In each case it is assumed that there is a unique and reproducible algorithm or data map that allows transformation from one view to an adjacent view. Most data management systems go through the motions of these transformations, although some may be combined. The converse thesis states that the extent to which they do not go through at least these transformations is the extent to which they miss having data independence. These five views of data will be examined in turn.

Stored Data Relations

Stored data relations or descriptions are that body of metadata describing the stored record as understood by the storage subsystem and hardware. This includes representations, device and hardware dependencies, access methods, among others. They are conceptually totally independent of any logical relations among the data items known by the data management system. One could construct a mapping procedure that says - 'My view of this data is that it is logically related in such a way, but I choose to store it in a totally different way'. This would probably lead to very inefficient operation, which of course is the reason which the logical data relations tend to be reflected in stored data

relations in today's systems. A stored record is a body of data known at the storage level. It may also be a unit of transfer. However, there is no necessary logical relation among the data elements implied by the stored data relations. Any reasonably efficient system and any reasonably intelligent data administrator may wish to reflect the logical data relations to some extent in the stored data relations. This is an efficiency and performance consideration - not a conceptual consideration.

System-known Logical Relations

At this point we have left any concept of relations among aggregates of stored data. We now consider a logical view of the data described in some data definition language. There are many ways in which these logical relations can be expressed, and much of the discussion in the literature is concerned with particular methods of expressing relations. In this paper we will consider them as sets and subsets. In this view, there are very general association capabilities between sets and subsets. It is possible to associate subsets of disjoint sets into yet another set. No logically consistent grouping of data items should be prohibited at this level. One may use these relations to access data. There is no logical requirement that one uses data relations to access data. These logical relations may be maintained automatically by the data base management system. They also may be maintained by an application program. It should again be emphasized that the set relations declared at this level may or may not be a useful model of reality.

Program-defined Logical Relations

These are the statements in a program which describe its view of the subset of data elements from the data base that it wishes to deal with at this point in time. It has sometimes been called the program's 'logical record' because of the carryover from traditional record oriented processing. At this level, the program is assumed to know what data elements it is potentially going to deal with. In this view of the program-defined logical relations, it is possible to statically declare the program's logical record groupings. Many programs are record oriented in that they will describe at this level a relation among named data elements by specific declaration. This declaration is independent of the algorithm that the program represents - that is, we prohibit a dynamically declared set of data relationships. For instance, suppose a program wishes to retrieve fields a, b and c in ascending order of b until all values are exhausted. At this level of declaration we explicitly prohibit dynamic name change such that the name 'c' is changed to 'd' to cause retrieval of a, b and d in order of b. Only a static declaration of the programs view of the data is allowed. This does not prohibit the temporary materialization of data elements which do not exist in the data base. It may also cause the materialization, for reasons of efficiency, of auxiliary paths to data which may exist only for the duration of the execution of that program. An example is the materialization of an index or some access path in the above case of retrieving in ascending order of b. This assumes a suitable algorithm for computing such materializations, either a system level algorithm or another application program which was dynamically invoked at that point.

Algorithmically Defined Relations

These relations are private to the current execution of a particular program. They depend upon the dynamic execution history of that program and therefore are inherently uncontrollable by a data base management system. The DBMS can probably find out such trivial things as changes to the data base but it has no understanding whatsoever of the data relations that were implied during the execution history of the program. One manifestation of the algorithmically defined relation is the logical/physical delete problem. When a record is deleted, is it physically deleted, or is only this program's knowledge of that record deleted, or is merely this activation's knowledge of that record deleted? Another manifestation is the use of REDEFINES in COBOL which can cause ambiguity. Yet another is the fact that FORTRAN input/output can be very dependent upon the

particular execution of the program.

User-Perceived Data Relations

This level is the mental relation synthesized by the end-user from the data presented to him by one or more application programs. It is totally unknown to the DBMS. It may not even be algorithmic in nature. However, it is not a data independence problem.

II SYMBOL RESOLUTION

The term Symbol Resolution is used in this paper to denote the process by which the mapping from one to another view of data occurs. The term 'binding' has been also used for this phenomenon, but 'binding' in many of today's systems is also ultimately connected with resource allocation and scheduling. Both of these are irrelevant to the process by which data names, data attributes, and data values are mapped or related through the several views of data.

Fundamental aspects of symbol resolution are:

1. Via a succession of mappings, it is necessary to define a unique association between the application program's unique name and description of a (set of) data element(s) and the hardware's unique name and description of the corresponding stored data element(s).
2. The point(s) at which this resolution, or any part of it, occurs is the major contentious issue in data independence.
3. Symbol resolution has parameters of both extent and time.

It is obviously necessary to develop a unique association between the program's definition of a data element and the stored data element that the program will access. Most of the discussion on data independence has concerned the point or points at which this resolution occurs. This has been confused by lack of recognition that symbol resolution has parameters of both extent and time. Thus, symbol resolution will occur at various points in the process, and it may occur to a variable extent at each definable point in the process. Symbol resolution involves not only name-to-address association, but also all the mappings, access paths, representations, and data attributes necessary to completely and uniquely satisfy the application program's data requirements.

In the case of materialized data items (virtual fields), it is entirely possible that the symbol resolution process is bi-directional, part of it originating from the data base toward the application program, part of it originating from the application program toward the data base. There is no inherent requirement that symbol resolution occurs undirectionally through the set of mappings, although this will be the most usual case.

The symbol resolution process does not have to occur uniformly in time. The process may be viewed as a matrix with time as one axis and item or attribute resolved as the other axis. It is possible to associate some attributes at compile time, some at load time, some at execution time, some at design time. This, of course, cannot be done in an inconsistent fashion, such as trying to associate the length of a data item before doing name association. However, there is a good deal of freedom possible in selecting when and what to associate. The decision of where and when to do this symbol resolution is the major factor in the successful implementation of data base systems and the application programs which use them.

Many of the arguments on data independence have been caused by the fact that symbol resolution is a major performance trade-off in today's hardware and software. This is not going to greatly improve. Early and more complete symbol resolution is more efficient of computer time than late and less complete symbol resolution. Many systems bind at compile time instead of execution time. In fact, much symbol resolution is done at idea conceptualization time and at design time. However, early symbol resolution is generally less efficient of people time, since flexibility is lost.

The demands of change in technology and business methods will force a gradual

migration to data base management systems which offer a range of symbol resolution options. These will include the ability selectively to resolve data attributes at various points and the converse, the option to dissociate these attributes. Thus it would be possible to move towards interpretive data access while the data base is being restructured and then to re-resolve the symbols to permit efficient operation with the restructured data base. Such a mechanism is within the state of the art, and would help to eliminate many of the compromises in today's systems.

Data independence requires the migration of symbol resolution from the compilers and application programs to the DBMS. To the extent this occurs in the DBMS, to the same extent will data independence be improved. Data independence is compromised if symbol resolution is performed algorithmically in the application program. Explicit program awareness of pointers, and chains, especially in the context of GET NEXT, is deleterious, as are the more subtle cases of knowledge of hierarchical relations in the data base: e.g.; the knowledge that a logical child is one level removed from the logical parent. Most of the application programs written today take explicit or implicit advantage of these data structures and will not operate properly if the data structure is changed.

Data independence is lost if symbol resolution is done by an application program algorithm which makes use of physical or stored data descriptions or relations. This appears to be the most widely felt user need in data base technology: the desire to change field length and representation, add or delete fields, change hardware-dependent attributes, without changing the application programs. This level of data independence is achievable at reasonable efficiency and without violating the structure of existing programming languages.

Data independence at present is achieved in one of two ways: either recompile the programs or dump and reload the data base. This is unacceptable for large installations now and is unacceptable in the long term for everyone. Independence from stored data relations and representation is both necessary and practical. The emphasis on machine efficiency is a direct result of the high apparent cost of hardware relative to the cost of people. As information systems become more pervasive and hardware costs decrease, this emphasis will change.

Specifiable time and extent of symbol resolution is a more complex requirement. It involves not just the DBMS but also operating systems and programming languages. A unified system wide view of symbol resolution is required for successful implementation.

DISCUSSION

GOSDEN:

One part of data independence is "portability". The major way in which users have achieved portability is to pretend that everything is an 80 column card and to keep all the symbol resolution in the application program. That conflicts with your last point. I believe you are basically right, but could you please amplify.

JARDINE:

If you have only one data format and only one representation and any other flexibility is prohibited, then there is no problem. A lot of mappings become one-to-one mappings; the only logical relations that you can define are those which are defined by the physical stored record, the only kind of data representation that you permit are holes in cards. You're then dealing with a very specific set of representations and a very specific data structure. If you prohibit any other then the problem goes away.

REINSTEIN:

I suggest that in the case John Gosden talks about, what you really have is

portable media and not in fact portable information. While it is easy to transmit tape or to carry 80 column cards from one place to another, it is much more difficult to get them from one program to another. If we really want to achieve information portability, a data base management system is essential.

LEFKOVITS:

John Gosden's question had to do with portability. What is the relation, if any, between data independence and portability?

JARDINE:

Data independence is the ability to restructure some or all of the data base in some ways without having to change the application programs that run against it. I don't know whether that permits one to take an application program from this data base system to that data base system and have it still work, or whether it allows one to take a data base and move it underneath a set of application programs without some change, but given that I had application programs that worked at some point in time, I want to be able to modify the structure of the data base without having to alter the application program.

SIBLEY:

I'm mildly disturbed at the definition that is coming out of data independence. Therefore, I'm not giving a view, I am trying to ask some questions. Data independence has several ramifications. The first is potentially to change the access method without changing the program. The second might be the change of storage structure. In other words we'll take the actual physical relationships and map them in some entirely different fashion. The third might be to change relationships in the data instances. Maybe some large subset of the programs would benefit substantially if I put in different relationships; i.e. throw out the old ones and put in new ones. As an example, suppose I have a data base on family relationships. Assume that we have father, son and grandson relationships. Obviously there is redundancy in this. We decide that instead, we'll just have father, son. However, we have programs already that ask us for grandfather. Therefore, we can write what I'll call a data base procedure to materialize grandfather as the father of the father. In other words the definition of the relation internal to the data base is a procedure. Don Jardine mentioned the idea of application programs having a relationship imbedded with them or in other words having some of the data structure associated with the program. I think we can have programs associated with the data base which allow us to have data independence.

Another topic I would like to discuss is the concept of "next" as in "get me the next". In one case it means that we have a predefined order, that we know how we write the structure out. It might equally mean that there is data of the same type which has an ordering defined in the data definition. It can be used in two different contexts. If you said, "Get me the next record", then presumably you don't know what sort of record you are getting. If however you said, "Get me the next person", then that is well-defined if you have an ordering on person stated explicitly in your data definition. Maybe "next" is not so dirty as it first seems as long as it is well enough qualified. The question in some of these problems is: how much qualification do we need?

Another set of problems arise as a result of changing the programs but not changing the data. Is this any sort of restriction? For example, suppose that my data base contains people and I ask, "Give me the dog belonging to Jack Jones". What should the machine do to me if I define a data base which consists only of people. The application program knows that at some future date we are going to include information about pets. We are writing a new program and we include in it the ability to ask for the names of particular sorts of pets - even though the data base doesn't contain it. Should we be allowed to do that or should the

compiler or the data base management system raise an error?

METAXIDES:

There is a difference between the data about pets not being in the data base and the description not being available.

SIBLEY:

I mean that there is no description in the data base about pets. However, I will put into my application program the fact that I want people and I want pets; then I will allow myself to write programs about it on the assumption that in the future the data base may contain it. In general, systems are not developed at the present time which allow us to ask for things which are not in the system.

MELTZER:

One situation is to be allowed to compile a program in preparation for information (including descriptors) later to be added to the data base. Another situation is to be allowed to compile and execute the program as long as a particular branch is never taken. This is quite different from the system saying, "I don't know". Perhaps if you have got a very sophisticated system the compiler might say "Be careful", but that doesn't mean I can't compile the program. Almost every system known today allows you to compile your programs in advance of the data being available.

SIBLEY:

Almost every one of them aborts immediately if you ask for it. I would expect it to come back and say, "There aren't any pets for this man in my data base".

METAXIDES:

There is surely a difference between the system knowing about the description and possibility of pets existing in the data base and whether there is any actual data. Provided the description exists, then clearly compilation is possible and it should be possible to run my program and ask whether there are any pets. I will get (hopefully) the right answer: 'There are none'. If the description has no indication of the possibility of pets, then it would be helpful if I was told that I am using references that simply are unknown, because it may clearly be a logic error at that point.

SIBLEY:

If we take the case of the CODASYL DBTG there is a definite statement under the DBTG statements that the subschema shall be a subset of the schema. I am suggesting that that is an unnecessary restriction.

METAXIDES:

The latest version says that Subschema stands not for subset but Subordinate Schema, meaning that the Subschema must be something that is derivable from the overall description of the data. In fact the April 1971 (CODASYL DBTG) specifications allowed Subschemas which were not true subsets of the schema.

SIBLEY:

I am saying that the Schema only says people and the Subschema says people and pets, where the overall DBMS knows nothing about pets whatsoever.

METAXIDES:

You have to decide whether you want to allow that kind of thing or not. As a user I would like to be warned that I am talking about data that the system knows nothing about and for which it has no descriptors.

SIBLEY:

Another aspect of data independence is changing the machine without changing the programs or data structure. If we have a DBTG representation on two machines should we be able to go across machine lines? Should we consider the standardization of a mixture of programming language and data structure? The final problem is that of changing the data base management system but not changing the data structure or the machine.

To summarize, let us assume some bulk of data with its definition. There are users with application programmes, there are application data definitions associated with the application programs, and there are certain accessing definitions associated with the data. The question is whether we can derive a mapping such that at some future time we can change relationsips, structuring, storage, and so on. The mapping functions are the way that the application program accesses from the data base and ultimately stored back again. The access definition is essentially all of the definitions of the accessing methodology, the storage structuring and presumably also the logical relationships within the data as stored. Data and the structuring of the data changes with time and consequently we must accommodate mapping changes. We introduce new programs with their own descriptions and their own mappings for accessing the data bases. This is one way of decoupling at least, data definition and storage definition from the application programmes.

CODD:

I would like to make a comment on one of Don Jardine's points that the stored representation of the data was conceptually independent of the logical. There is a constraint in there that we cannot overlook and that throws a monkey wrench into a lot of discussions in this field. The operations that you perform on the logical view of the data have to be capable of being faithfully simulated on the stored representation of the data. This applies not just between these two levels, but any other levels you care to put in. Whatever operations have been specified at one level have to be faithfully reproducible in some way at another. There's been a good deal of loose talk about providing the users with subschemas or their own views that can depart from the main logical view, the main schema, radically: say, providing tree structures at one level and non-hierarchical structures at another. There are serious problems of reproducing insert, update and delete operations at one level, in another level. Now, I think the statement has to be modified by that constraint.

JARDINE:

Many data management systems currently in existence require that there be nearly one-to-one mapping from the logical data structure to the physical data structure. My point is that this requirement compromises data independence. I agree entirely that I've got to get there uniquely and reproducibly.

CODD:

This implies to me that it's statically possible to map from the user's names and descriptions to the system's names and descriptions. There is more to it than mapping the names, the operations have to be capable of being mapped. You can get two radically different structures being equivalent from the standpoint of query but not equivalent in behaviour from the standpoint of insert, update and delete. In one of your examples you implied that the user merely had

to select any subset of the fields to arrive at his own scheme. In fact that won't do. Let's take just two fields, supplier number and part number. It is a very different thing to extract supplier number and part number in the context of 'such and such a supplier is capable of supplying a part' versus the context 'he is currently supplying that part'. In other words, it's not enough to state at any time just the fields; you have to be explicit about the relation in the context of which you are specifying it. That does not mean to say that the only relations you can talk about in the user's view are the relations that are explicitly defined in the system's logical view. You can also talk about things that are derivable from this.

LEFKOVITS:

At this point we have a problem we have not been able to define and we are trying to make vast generalizations about it. Let me suggest an approach. Suppose we have a program that accesses a data base, or if you wish a set of programs, which is written in some well defined programming language. Now suppose we enumerate all of the entities which are present in the system, both hardware and software, which this program touches or utilizes. This can get to be a very lengthy list and we may want to decide that we really don't want to bother about the way you represent a byte in memory or what have you. Assume we have a list of entities which are present in this system. Suppose that we define grades or levels of data independence. The number of factors here is obviously very large. This is a multidimensional space and we cannot hope to cover all possibilities. However, we can pick on some that seem to be interesting and define the level of data independence that is achieved.

JARDINE:

If we look at the kinds of changes that people want to make to data bases without impacting the existing application program, we are constantly concerned with such things as adding new fields to the data base, with changing the length of fields in the data base (not the represented lengths but the actual lengths), the number of digits or characters. There is, I think, a small list of items that constitute 80 per cent of the trauma.

METAXIDES:

First of all, I recommend to your attention two papers that appeared on this subject in the Proceedings of the 1971 ACM Sigfidet Workshop. They give a clear picture of what data independence is all about. The papers describe logical data independence, meaning the insulation of the application program from changes to the description of the data and associated changes to the data base itself, or to look at it another way, the ability to vary the data base structure without affecting the ability of a program to give the same results that it did prior to that change.

One of the papers does what Henry Lefkovits has just suggested; it lists the reasons for various changes and it discusses whether or not logical data independence can be achieved. One of the changes listed is adding a new field to a record of interest; associated with that is changing the length of the field or its data type.

The next change listed is deleting a field of interest. There is no way you can be independent of that, thus total data independence is not achievable. The third change discussed is adding a new record entity to the data base. Can the program still work? Another one is moving a field of interest from one record of interest to another record of interest. All of these can, I think, be achieved with the exception of the deletion.

Other, more tricky situations are those in which we have, for example, assumed the existence of a one-to-one relationships between two entities where in fact there is a one-to-many relationship. It is impossible that such situations can also be dealt with but they are far more difficult.

DATE:

I'm one of the authors Tax Metaxides was talking about. I'm grateful to him for this unexpected limelight. I think that it might be useful as a starting point to take these two things as some sort of working document. I would add though that they were written over two years ago and I certainly have reservations about some points in them. In fact, what I would like to do is go away and rethink it and make some changes, but not major changes.

WRIGHT:

With reference to the different kinds of data relationships that were known, the one that always caused me the most trouble was the relationship that was there and existed but I didn't know about it at the time. There are a lot of these kinds of relationships that exist in the data, that aren't known at the time the data is collected and recorded and that must be used at some later time. It seems to me that these are a key kind of relationship which we should all keep in mind when we're talking about data independence.

LOWENTHAL:

Tax Metaxides mentioned that one of the more difficult problems in data independence, for example, was going from a one-to-one relationship to a one-to-many relationship when you find out the real world has been modelled incorrectly. I maintain that that kind of data independence is impossible. The object of trying to achieve data independence is not to make up for mistakes in modelling the real world. For instance, one might represent a hierarchy as a flat structure by replicating the data and then decide that he wants to make a hierarchy out of it. That should be transparent to the user program. But if he somehow finds out that the real world is a network, you're not going to get by without changing the user program. You might think that you're getting by without recompiling or restructuring the application program but that would probably result in incorrect answers. In discussing those specific issues in data independence, we should aim toward reconfigurations of what was essentially correct information.

LIGHTFOOT:

Over the last few days I've been listening to discussion on data independence and I began to consider what will it cost a running production environment. I came up with some rather strange figures that disagree with a lot of people here. In seven years on restructuring of data bases we have spent $20,000. How much overhead would be added for data independency to save that $20,000? Five per cent, ten per cent,? Fifty per cent? Five per cent of my budget alone would be $50,000 a year. I'm concerned it could be so expensive that it could make processing prohibitive.

KIRSHENBAUM:

The point is that not everybody gets it. There are some users and some applications that require this kind of facility. What we're really talking about is the set of choices. You should be able to say: 'compile and bind' if that's appropriate to your application. That presumably would be more efficient than waiting until each access. Some applications, particularly utilities or report writers, must wait until much later in time to bind the data to the program.

HILL:

I feel that data independence is a very powerful and very desirable thing. There is not a system on the market today, IMS included, that will provide what

you are asking for. We've been talking pretty much about data base management system with total exclusion of the areas of data communications. I think that the two go hand in hand, that we really ought to be talking about the data management system. Whether it's stored data, whether it's transmitted data, whether it's data from a terminal, you have the same sort of problem for data independence.

METAXIDES:

In a data base environment we are gradually building a data base of larger and larger scope. More and more programs are written that interact with that data and they affect more and more aspects of our organization. Each time we make changes to the data base we do not want to be forced to rewrite programs. As time goes on I may have 200 or even 2,000 programs. The point is that I would not wish to rewrite those programs as my data expands in scope or when it becomes necessary to change its description.

SIBLEY:

I heard three speakers on Monday telling me that data independence was one of the reasons they went to data base management. I'm sure that they all didn't mean the same thing. There is a substantial difference between the amount of data independence that is meant by people saying 'data independence' and I would like to pin it down once and for all. Maybe if we can define what we mean by it we can say how much we need.

SHEEHAN:

I think most of us in this room understand what data independence is, at least insofar as we have to cope with it at this moment. It's true that there seems to be great difference between your definition and Ted Codd's definition. The only trouble is that while data independence is still being defined, we have to deliver a system. I don't know what those data bases are truly going to look like in a year and I can't stop development on several projects. This kind of a discussion has to be split in terms of what do we mean in the long-term, and what do we mean short-term to carry us smoothly from the present so that when things are available we can make use of them.

BASH:

Data independence to me is a relatively short-term thing and I'm sure it won't satsify some of Ed Sibley's needs but it may give some examples. I know I'm going to make mistakes. Data independence may provide me an alternative to recover from that mistake other than reprogram. Within that scope in many ways I think the logical data base concept of IMS provides the first level and much of the CODASYL Schema and Subschema does also. Mappings have to be reversible. In the mathematical sense a mapping must be a function, because it's very easy to map from the data base to a program but what do we do when the program requests a delete? I have to be able to reverse that mapping correctly. Whatever we do here in providing these mappings, they have to have some sort of very clearly defined reversibilities. In the short-term we need, in IMS terminology, the capability to alter the order that GET NEXT provides things in. We need to be able to take a segment or logical chunk of data presented to the program, and break it into two. We also need to be able to go field shuffling within a segment. In addition, we need to be able to add a segment within the data base without going through a complete reload.

JARDINE:

What we've been hearing is a strong desire to have the program insulated

from changes in the stored data relations and representations, and in some cases but far fewer, from changes in the logical relations among the data, either at the system knowledge level or at the program defined level. The system-known data relations and program-known data relations are, in most cases, some kind of model of the real world which changes much less frequently than our representation of data in a particular set of hardware and software. Changes in the system-known data relations and program-known data relations are going to be as a result of changes in the way in which the particular enterprise carries out its business activities. Companies reorganize and change frequently, but the way they carry out their business doesn't change nearly as frequently. However, representations, storage devices, and software change all the time. The emphasis has to be in the stored relations. The rest of it can be handled. It's not pleasant, but I think more people are willing to put up with it. At least the reason for the changes in the system-known relations and the program-known relations have been caused by the enterprise itself and not as a result of some obscure activity on the part of the vendor.

EDITOR'S NOTE:

D. Moerke raised the problem of vendor independence and the extent to which data independence assisted in it.

METAXIDES:

The question of vendor independence is a little different from what we've been discussing but there are certainly trends in that direction. Common languages are one step, but even if common data base languages are implemented by the various vendors, there will not be complete portability between vendors because we still have the problem of data compatibility. It is always possible to unload from one data base implementation, say of the DBTG type, and load to somebody else's, but that is a big endeavour. Insofar as the same data base management system languages have been implemented by various vendors, there is a possible architecture that will give you independence of the vendor on which your applications are placed. This architecture involves the inclusion of a data base machine. In line with the comments this morning about the economics of distributed processing, it would be possible to develop a data base machine which services all requests for data and handles all interaction with the data base. The data base machine could interface one to n host machines where the application programs reside. Now you are dependent on the data base machine so perhaps you've just moved one step sideways; but at least you would be able with this architecture, to have concurrently interacting with the same data base, programs that are running on different host machines from different vendors.

TURNER:

There is a project, part of the ARPA network, called the data computer. It's a stand-alone computer with its peripheral storage device that's hooked on to the network. Processes running on the machines, which are hooked to the network, can make requests of the data computer and obtain data back.

Data Base Management Systems, D. A. Jardine (ed.).

INFORMATION, MANAGEMENT AND THE STATUS QUO: Some Organizational and Social Implications of Data Based Management Systems

IVAR BERG
Columbia University
New York, New York

The most natural inclination, upon receipt of the invitation to prepare a paper for this symposium, was to organize some thoughtful remarks about the much vexed issues of personal privacy that have come to the fore in the context of our growing capacities to gather, organize and "process" data on people. A review of what turns out to be a burgeoning literature on that tortured subject persuades me that there is little I can add, at this time, to the intriguing and ongoing discussions.

A reading of this literature, however, has also persuaded me that there are other important organizational and social implications of data based management systems (DBMSs hereafter) that have been almost totally ignored, and it is to some of these that I have elected to address myself. The discussion is in three parts. First, I should like to introduce and briefly develop a useful distinction that highlights two interrelated dimensions of organizations. Next I will try to trace, in outline form, some of the consequences of DBMSs for each of the two dimensions of organizations and, finally, in the third section, I will attempt to specify the most readily predictable implications for efficiency and equity, in American society, of the organizational changes outlined in the second part. My mode of exposition, as this introduction may suggest, requires that I beg that reader's indulgence for what I hope will only briefly appear to be a rather elliptical path to the central points of the paper.

1. ORGANIZATIONS AS "RATIONAL" AND "NATURAL SYSTEMS"

Social scientists have long been concerned with the implications for the structure and functioning of both organizations and their host societies of major new developments in the design, the staffing and in other significant characteristics of organizations. Thus, such "watersheds" as the shift from domestic industries--the so-called putting out system--to the factory system, the advent of the separation of ownership from control in modern Western corporations, the introduction of collective bargaining, and the development of double-entry bookkeeping are just a few among the innumerable developments that have provoked lengthy theoretical discussions and empirical investigations designed to clarify the meanings and consequences of quantum changes affecting the managers and members of organizations, and the larger social systems in which their organizations are embedded. My professional sociological assessments and my experiences in management lead me to the conclusion that DBMSs are among such quantum changes.

It is only a little unfair, in summarizing the large accumulation of scholarship to which I alluded in the previous paragraph, to lump the efforts under one of three headings. Under the first we may place all the studies that deal with watershed developments largely in terms of the formal or rational structure of organizations. Under the second we may place all the studies in which emphasis is placed upon the less planned, less engineered and more emergent social systems found in all organizations. Under the third, and by far the least populated rubric, we may place those discussions and investigations that deal with organizations as simultaneously "rational" and "natural" systems, in which the interplay of formal structure with "informal" systems is stressed. The dis-

tinction between organizations viewed as emergent social systems and organizations viewed as rational systems, I must stress, is one of emphasis and is not intended to imply a dichotomy or a pair of mutually exclusive categories: Investigators have tended to focus on one or the other of a two-dimensional conception of organizations in their efforts to assess the impacts of various changes in social and organizational "technologies," and the analytical distinction has been with us, implicitly when not explicitly, since social scientists first began, in the period after the French revolution, to look, in depth, at social organization and the strategies for studying and improving them.[1] Thus Henri de St. Simon, in the early 19th century, first emphasized the superiority of planned, rational social organizations, whose design would be informed by scientific principles, over those whose logic was informed by the principles of hereditary aristocracy. One of his leading students, Auguste Comte, fought what we regard today as social engineering, and stressed the superiority and durability of organizations whose principle component parts emerged "voluntaristically", that is, whose arrangements grew out of the interactions among people and whose stability could be linked to their emergent qualities. In more modern times we can easily discern the residues of what was, early on, a matter for dispute. An intelligent review of the literature on organizations will suggest that some theoreticians and investigators, while avoiding conflicts over the question, will tend to stress the formal, bureaucratic aspects of organization and the variances in organizational behavior patterns or in respect of organizational efficiency that are attributable to variations in the structural aspects of organizations. Variable spans of control, variable numbers of levels of hierarchies (i.e. whether an organization is "tall" or "flat"), "close" or "distant" supervisory arrangements, variable rules and countless similar structurally given, rationally intended attributes are thus singled out for study, and the concern is with discovering the correlates of these variable attributes.

Other investigators, while conceding the importance of "formal organizational structure", tend to emphasize the social systems, the spontaneous and voluntaristic norms, understandings, rituals, punishment and reward systems, the working rules and accommodations that grow, naturally, out of the interactions of people in different levels of organizations and out of the interactions among peers and between laterally situated persons.

The first "school" thus homes in, for analytical, diagnostic and perscriptive purposes, upon the formal rules, the reporting channels, the "SOPs", the routinized conventions described in procedure manuals, and upon a host of similar structural attributes that specify, design, order and shape the behavior of people. This collection of relatively stable arrangements--seniority rules, personnel procedures, accounting schemes, work standards, job descriptions, protocol arrangements, to add a few to those already mentioned, we may refer to as the "rational system". The regulations, procedures and modus operandi that are thus standardized, publicized and explicitly held to be binding on organizational members, are conceived to be rationally adapted (and adaptable) to legitimate organizational ends; for the most part they go substantially unquestioned.

The second school, comprised of human relationists, recognizes the often useful, purposive opportunities and constraints implied by formal structural components, but they emphasize that no manager, however perceptive, can design all that is needed by way of influencing people, and they attend to the inventive "ententes", the quid pro quo bargaining, the "unofficial" but full-blown social systems that are developed to contend with the inevitably incomplete quality of organizational structure.

We need not pause here to review the now abundant human relations literature to make the point that these "natural" systems may be both constructive and

1. This part of my discussion is very heavily influenced by a seminal article by Alvin Gouldner of Washington University in St. Louis entitled "Organizational Analysis" which was published in R. K. Metron et al (eds) Sociology Today, (N.Y., Basic Books, 1959), pp. 423 ff.

subversive in their effect upon organizations--in my own experience these systems are a mixed blessing. The more immediately relevant observation is simply that both the "rational" and "natural" systems exist, and both are ammenable to some degree, to exploitation and study.

It is also highly relevant, in the present context, to point out that the "natural system, so called, the spontaneous and emergent social apparatus, thrives on many realities the most immediately significant of which

1. inhere in the interdependencies of people and groups with related and conflicting organizational obligations;
2. the high degree of uncertainty associated with so many aspects of organizational life (the future, the "market", the tenure of others, the moods of leaders, crises, the unavailability of various kinds of information, the reactions to ever-changing stimuli, etc.);
3. the incomplete quality of any organization, which after all, is facing ever-changing contingencies and confronting changed exigencies; and, finally
4. the obsolete attributes of organizations that have survived despite changed circumstances.

These realities, among many that could be enumerated, lead to bargains, implicit accommodations and to other of the "natural system's" configuration. Among the major values at stake in these quid pro quo arrangements are those attaching to information, especially information relevent to the organization's operations.

Moving to the organization's boundaries we may confidently assert that the same two systems have their play in the relationships between an organization and the agencies "outside" the organization. Thus, there are formal rules of the game--legal, contractual, governmental etc.-- that color relationships between the organization, its clients and regulators. But, once again, there are also implicit arrangements that help lubricate the bureaucratic machinery, just as within organizations there are informal or "natural" systems coexistent, indeed coextensive, with "rational" systems.

2. DBMSs, THE "RATIONAL" AND THE "NATURAL" ORGANIZATIONAL SYSTEMS

Having said all this, rather briefly, about the "rational" and the "natural" systems it becomes necessary only to assert that it is one of management's great challenges to balance the two "systems" over which it presides. It is my thesis that DBMSs help managers to strike this balance better though, as we will see, such prospects may put some managers "on the spot"!

The reader will recall that, in the earlier section, much was made of the role of informational shortcomings, and of (related) uncertainties, in creating the forces out of which the adaptational or "natural" system emerges. DBMSs, operated by permanent units akin to the comptroller's apparatus,[2] would very likely have the effect of reducing, by a very substantial magnitude, the number of "negotiable" issues around which persons and groups reach the agreements and ententes to which I have referred. DBMSs thus would extend, by a good bit, the "rational" system; it would reduce the number of issues on which judgments, in the absence of adequate information, have to be made in accordance with the norms and conventions that inform while hosts of organizational judgments.

On the other side, the influence, the "play", of the natural system would be reduced to the degree that the rational system's extension (via well managed

2. I conceive of DBMSs as stable units containing data processing skills, statistical skills and either representations or actual representatives of the functional skills needed by the organization such as marketing, production, surgery, academic planning, economic forecasting or other specific "fields" germane to the firm, university or other enterprise. I also assume that common definitions, codes and "time frames" are among the "impositions" DBMSs would seek continually to define, expand and apply to data on people, processes, methods and resource utilization.

DBMSs) denies the necessity for "adjustments".

A few illustrations may serve to drive the point home. Thus, the absence of what is, nowadays, conceived to be a well-functioning DBM system at IBM figured significantly in the capacity of a number of that company's scientists and engineers to "baloon squeeze", i.e. to expend resources over a very long period for other-than-budgeted operations. Happily for IBM these employees managed, ultimately, to produce the very core memory on which top management had given up, through the illegal resource expenditures they continued to make after top management issued "stop" orders. One may only marvel at the successful quality of the natural system that managed, for what Mr. Watson now confesses was a long time, to obscure the realities that became known only after the experimenters had been successful!

Consider also that more than 20% of the firms, in a study of subcontracting decisions in the '60s, told investigators that they did not have information on which to make all-important "make-or-buy" decisions, comparative information, that is, on productivity, costs, schedules, etc. Given the well-known attitudes of unions towards subcontracting, one has little sympathy for managers' assertions of "management prerogatives" under such conditions of their exercise.

And a very large Long Island, N.Y. government contractor decided to subcontract all of its design engineering until its engineers, on their own time and with the aid of rented computer time, were able to do a crash study of the comparative costs that management had overlooked for lack of a sophisticated management data system.

Economists, meantime, in a study a few years ago of pricing "decisions" made by a large number of blue-chip firms, discovered that, in nearly one quarter of the firms, no one even knew who set their prices or how these prices had been calculated!

Government regulators, meantime, agreed with ATT, about three years ago, that it would cost hundreds of millions of dollars to do the type of study necessary to establish, empirically, the costs of the activities related to services for which rates have to by set. So far as is known a crude estimating procedure is utilized for a host of cost decisions.

And one of the largest heavy electric equipment manufacturing companies may never discover that the managers of one of the largest units in this decentralized corporation "smooth out" profit curves so that headquarters will not, after a "big" year, elevate its expectations of the decentralized unit and its managers. Among the helpful and undiscovered devices: Annually variable contributions to community fund activities!

Next, I may mention two 1972 Labor Department studies, both involving manpower planning. In the first study, of a very large number of firms, researchers found that almost none of the firms ever traced the manpower implications of marketing, financial or production decisions, despite the fact that there was good evidence, among the minority of companies, that major manpower headaches could, by better collection and analysis of available planning information, have been avoided. DBMSs, obviously, can be designed to make just such collations a matter of regular corporate routine.

In the second study it was found that almost none of the employers, in another large sample, ever examined the organizational relevance of job requirements utilized in the screening of any of their employees. My own recent studies give abundant corroborative evidence[3] to these Labor Department findings. Once again, DBMS can easily include designs for realistic assessment and review of selection and pay problems, a fact of great moment given the very large portion of costs that relate to manpower.

I should emphasize here that these illustrations are offered not as atrocity stories but as evidence that many of us manage in obsolete ways when we do not

3. See the author's Education and Jobs: The Great Training Robbery, N.Y. Praeger, 1970.

act upon the basis of a decision matrix in which we seek constantly to at least recognize, if not expand, the portions of our judgments that are "data based".

DBMSs as I conceive them, meanwhile, reduce the legitimacy of two types of familiar management complaints. A good DBMS unit will constantly be working to develop measurements where none exist, and to refine those that do exist. The inevitable decision-making climate, under such circumstances, becomes increasingly inhospitable toward the manager who claims that he "would make better judgments if he had better measures and more information". That often heard complaint is a far less defensible one in an organization in which hunches, agreements, cover-ups fudged reports and crude estimates are confronted with a well managed flow of coherant information.

Other managers, those who are gluttons for hard information, often complain, in the absence of DBMSs, that their "conclusions would involve fewer value judgments if only they had more information". My own experience, in that regard, is that the more hard information the manager has under control, the more valid statistical information he has available, the more he comes to recognize that the basic decisions he must make do and always will involve value and not "empirical" judgments. Rarely are the parametric dimensions of a decision the most determinative or dispositive aspects of problem-solving. Well developed DBMSs will, however, help to clarify the value issues that the manager needs to join in reaching a final conclusion.

A DBM system could, by way of concluding this section, contribute to the following organizational developments, all of which would extend and expand the "rational system" dimension of organizations:

1. centralization of decision-making
2. institutional research (more reports, more procedures, more rules)
3. expansions in common definitions of a variety of variables that, in the absence of DBMSs, are defined differently by different subunits in the absence of a DBM system.
4. an increase in statistically over "clinically" informed judgments (more judgments based on evidence at expense of ad hoc procedures and standards)
5. proliferations of activity indexes of various kinds, bearing e.g. upon performance and resource utilization, made possible by increased measurements.
6. spread of commonality in "time frames" used by diverse organizational agencies such that different units and subsystems are not operating on different planning and other horizons.

In each case the margins for bargaining, for trading off, in short for the play of informal arrangements, are reduced. But, unlike formal, rational systems before the advent of DBM, the emergent bureaucratic or rational system, itself, would likely be less often rooted in capriciousness or whim than on increasingly hard information. It is a good deal less easy, in the face of expanded information, to defend a bureaucratic, rational procedure which cannot pass the tests of accuracy and "informedness". Thus DBMSs, I am arguing, will, by their emphasis on the role of information (whether it be quantitative or qualitative information) reinforce, and in a positive, constructive way, improve the rational system. And it is the rational system that, for lack of just such credibility as information provides, evokes so many of the responses we stereotypically identify with "insufficient, rule-bound bureaucracies". The rational structure's integrity may thus be enhanced and its pre-DBMS consequences reduced.[4] A manager, in short, will be hard put to defend a long-standing practice, an established procedure, or an entrenched policy that has resulted from a good hunch once the pressures of confronting tell-tale measures are a regular part of the decision-maker's occupational culture.

At the same time, the rationale for the natural system, typically couched in terms of the need to compensate for, or to flesh out, the rational system's gaps,

4. Needless to say, the effects of all organizational innovations are largely shaped by the conditions and circumstances of their introduction and implementation, a very vexing matter which, for present purposes only, I take as amenable to sensible influence and good management.

would likely be challenged by the output made possible by DBMS. The trade offs, the bargains, the understandings that are so important a part of the natural system become less defensible when the activities of the people and the units they serve become amenable to increased control-through-information.

Whether managers would really prefer such expansions in the rational systems they employ in their organizations remains, of course, to be seen. Managers who are sufficiently protected from the chill winds of competitive forces by various shelters may well not wish to give up "ad hocracy" for improved bureaucracy. And it is convenient, in many settings, to protect decisions from the types of empirical tests that DBMSs make it possible to devise. Finally, the public's interest in efficiency and equity is, as we will note in the next and concluding section, far more readily pursued in the regulatory process when somewhat more of the detailed operations of an organization are amenable to precise assessment.

3. THE SOCIAL IMPLICATIONS OF DATA BASE MANAGEMENT SYSTEMS

Laws, court decisions and commonly held values all point to the public's interest in an efficient, equitable social system. Inefficiency, whether in universities, foundations, unions, corporations or in public school systems, is costly as everyone knows. At the same time, there is growing concern about the rights of individuals. As the previous sections have indicated I regard the advent of DBMSs with considerable enthusiasm and in no small part because I can readily see that their widespread incorporation in organizations would enhance efficiency and extend the reach of equity.

The quest for efficiency, for a starter, is of considerable moment, as we see inflation in our economy, significant changes in our international trade relationships and widespread popular misgivings about the quality of management in both private and public sectors of our society. The point need not be belabored that to the important extent DBMSs facilitate more rational management they will help to improve organizational performance. I may mention, in passing, that the single largest complaint of employed Americans about their workaday lives, according to a very recent 1972 study of University of Michigan investigators, in that they have difficulties doing their work because of faulty scheduling, unavailability of component parts and tools and because of careless management[½] The detailed specifications of these complaints, by white collar no less than blue collar workers, point to the fact that there are many things to be done in the organizational worksetting of types to which DBMSs are highly relevant.

On the equity side there are no better illustrations of the potential of DBMSs than those that have emerged in connection with affirmative action programs. At the heart of the legal affirmative requirements are those that compel employers to specify job requirements, to justify these job requirements empirically, and to set goals based on relevant "outside" labor market information. A DBMS makes such specifications, justifications and comparisons a simple matter indeed. The resulting gains to management, as the Labor Department studies, cited above, clearly indicate, are considerable, almost as large, in fact, as the boon such legal requirements generate for equal opportunity.

In respect of both efficiency and equity, however, we may assert (with little fear of contradiction), that the public and private advantages are literally dwarfed by vested interests. The corrolary: DBMSs and other arrangements that facilitate the rational pursuit of efficiency and the democratic pursuit of equality are likely to evoke the same negative responses that have clearly been evoked by the pursuits themselves.

At this point we may return to the discussion of the "rational" and "natural" systems with which we began in the early pages of this study. The point is that the "natural system serves the organization only when there is no conflict with the organization's imperatives and those of self-serving individuals and groups in the organization. These persons and groups are self indulgent, self protective and apprehensive about their status and zealous in their guardianship over

resources, especially informational resources, the possession of which ensures their individual and collective security.

DBMSs, meantime, extend the rational system, which is one kind of threat to the natural system, even as they challenge the rationale, discussed earlier, for the natural system's very existence.

Let there be no mistake, in sum, that members of intra-and interorganizational "natural" systems will resist DBMSs, the effect of which are efficiency and equalitarianism! As the illustrations in part 2 may have suggested, we live and work in organizations, and in an economy, in which informal, emergent, voluntaristic "natural" systems are a fundamental part of a universal dedication to the proposition that

1. significant change is problematic and probably dangerous;
2. that competition is good "for the other guy";
3. that shelter-seeking is inevitable.

Even threats to the very existence of the organization can be assimilated by the natural systems that emerge within them. I can report that one year after a temporary contract debarment, involving millions of government dollars, was applied to a well-known organization for lack of an adequate affirmative action program, the DBMS developed to assure equal employment opportunity in that organization was totally and effectually subverted by the very nearly absolute dedication of the opponents of affirmative action in that organization. Sad to say, the subversive effort was facilitated by top management groups in this organization, members of which were first in line in the effort to "neutralize" an unpopular regulatory measure. Readers familiar with anti-trust history, with political campaign reform, with internal revenue, with import-export licensing, and with other devices serving the economy or its human operatives may add their own favorite illustrations.

Natural systems, whether within or among organizations, are not accidents; they are tenacious "supplements" in an imperfect world that facilitate the efforts of people to cope with demands. Among their major progenitors are the ambiguities in informational "monopolies" that DBMS could go far to reduce. The consequent opportunities for management that inhere in DBMSs, however, would tend to be cursed more than blessed by many managers indeed who, in their own way, have an enormous investment in the status quo, and in sustaining the conditions that force uninformed decision making. Information tests, like market tests, are by no means easily incorporated into decision-making styles that emphasize intuition. The social and organizational implications of DBMSs, in fine, are revolutionary, but there is little evidence that today's managers are revolutionaries or that our social system's members are prepared to swap their gentlemen's agreements, their occupational monopolies or other "shelters", for more equity or more efficiency.

DISCUSSION

MAYNARD:

Did you come to any conclusions on male versus femal employees, particularly in the professional area?

BERG:

The findings regarding performance and rewards that I cited, applied even more to women. As we know, they are also discriminated against in occupations, and in regard to salary and so on. But the same general findings occurred. If you compare better-with less-educated women in a given job category, the former were less adequate from a performance point of view. By the way, the performance measures involved were all management measures. I did not make an evaluation of whether somebody was performing adequately. I had managers' data on their

employees and managers' assessments of employee performance. I didn't invent the "dependent variable". The results applied to women as well as men.

MAYNARD:

It seems to me the skill level of our profession generally relates to the rational engineering mentality. Do you see a mismatch in the skills perceptions of our profession towards the rational system versus the natural?

BERG:

I don't think that the absence of computer scientists in top management has so much to do with differences in skill levels or whether this fraternity has less developed interpersonal skills or fewer anthropological sensitivities to the social life of the organization and the organization's social-cultural group structures. I think that what more likely happens is that career ladders are very different. This is not a fraternity, like marketing, finance, and accounting which are among the three major ladders to top management. My guess is, that changes are going to occur soon enough. I think that you will acquit yourselves as handsomely (and we hope even more effectively) than the marketing, accounting, and finance people who now populate management. I don't think that it involves educational differences or the fact that you have had, by implication, some kind of truncated training, that you are more heavy handed in manipulating social systems, or incompetent in dealing with the "rational system". I would guess that more of what you do for a living will be folded into management education so that the business fraternity will have a better understanding of the kinds of things that go on in marketing, accounting, and finance through your contributions.

I think that there are ladders that "don't go" to management in 1973, and ladders that do. The ladders that do are "monopolistically protected" by the people that do climb them. Marketing people will look back on <u>that</u> part of the business leadership as a major ingredient, for example. I would not expect that that would be your professional plight. The first plight this fraternity has is getting data base management units high enough in the organization and close enough to top management so that top management begins to get the blistering that comes from the confrontations with the information that could be generated in DBMSs in organizations that are less than rationally operated.

STEEL:

I think you are telling us we are doing something to institutions and organizations that might be roughly equivalent to the introduction of the principles of accounting in thirteenth century Florence; that there is an underlying profound change in the organizations as a consequence of this, and that almost nobody really recognizes it or appreciates it yet, and that it may be a generation before the organizations grow up to accommodate it.

BERG:

If I correctly understand what you are "up to" from your discussion of data base management, I see a potential water-shed that I would list with double entry bookkeeping and the relocation of the cottage industry. Whether the changes I see materialize depends on a lot of different things. One of the reasons your role will increase is the collossal expansion of the public service. It's the public services that are having the biggest exposures to these kinds of logic. It is no accident that when data base management systems are introduced they tend to be introduced in the public sector two-to-one over private industry, I would guess, in part for government contracting reasons and the consequent need to make better cost estimates. Because of the expansion of the public service, which doesn't have a "bottom line", the need, in short, for those of us in the not-for-profit sector to develop measures of performance against which to estimate

alternatives makes us more dependent on your technology. We have already discovered the need in higher education, as those eight or ten of you from universities in the room can recognize. In the absence of a "market test" we are forced into "something else" that makes it possible to have an evaluation process. Simultaneously, the economy no longer functions as it did in the early stages of the industrial era. It is a mixture of classically competitive and all kinds of other ingredients. We are, in addition, not really getting a full market test of the performance of private industry: Within private industry what is really growing is your overhead! We lost 500,000 manufacturing jobs in the last three years, but we gained over 2 1/2 million white collar-administrator jobs. The ratio of white collar to blue collar has changed dramatically and we have never, in private industry, been able to link very accurately the productivity of overhead to the productivity of production personnel. That raises internal problems in private industry that go well beyond just accounting, in determining where it is all going to end. There are no patterns, except what we might call "managerial discretion", in describing the white-to-blue-collar ratio in private industry. The ratios don't vary with anything systematically except what one could call management's highly idiosyncratic efforts to judge what its market or environmental challenges are. Thus, you have firms with enormous ratios and firms with very low ratios. An economy that doesn't test fully adequately the performance of firms and an economic theory that makes no room for external costs, really presses upon us the need for information systems on the basis of which we can make some estimates of performance, cost, and benefits. What you are trying to design in multiple software systems imposed on changing hardware systems is a set of logics that extend the reaches of economic science and administrative science simultaneously. Both are now paralyzed for lack of the kind of information they can get from data base systems. Managers simply don't know why they are doing many of the things they do, and the market isn't testing all the parts of their decisions. The market is testing (when it does test anything) the whole decision.

STEEL:

In the insurance industry it can at least be augued that our entire operation is overhead. It has some frightening implications.

BERG:

You have a little bit of help there. In light moments I point out that like bankers, "you can pay your help out of inventory". You also have another advantage, something that informal systems will continue to protect for a very long time: your actuarial bases are typically twenty or more years behind the Census. So even if we make you more efficient, you are still sheltered against the chill winds of the market place by a very substantial factor.

OCHEL:

One of the implications or basic facets of data management is that we are providing the users of data more control over the data. One of the trends that I am seeing is that in functional organizations, the better control of data tends to cause a more autocratic control of that organization versus a democratic control. How do you think this will impact the social structure of our government versus that of a commercial organization?

BERG:

I don't really sense that it would have a really tremendous effect on democracy. In fact it might help democracy a good bit. One of the capacities that the age of computers presumably offers the American people, and other societies, is a more responsive, representative democratic form, where votes and

tastes and preferences could be rendered more frequently instead of less frequently, in guiding the decisions. What do we have now? We have assurances from governments claiming to be democratic. e.g. "The telegrams are three-to-one in favour of my program". Whose telegrams? The evidence that we have in the last six to ten weeks (the Watergate inquiry) raises a serious question about where those telegrams came from and who paid for them.

I think that the threats to democracy are better gleaned from the writings from the likes of Thomas Jefferson, Madison, and Hamilton and the others from the Federalist Papers. I don't think that more information, per se, would have any necessary implications beyond improving the capacities of honest people to govern honestly, and for democratic people to govern more democratically. If, however, we have generations of cynical people who don't want to govern democratically, and who want to use information in undemocratic ways, or who don't want to use information but want to be inefficient, we have problems that go way beyond that which a fraternity like this can professionally "contribute" to the world. Those, I think, are other issues. I think the question of availability or unavailability of information is a marginal issue compared to this.

As I said, a question of privacy does arise and, clearly, management, when it fully gleans the implications of data base systems, may conceivably be opposed because public regulations are easier to enforce. The more kinds of data I could imagine (leaving aside personnel data), the more adequate information that the firm can have, the more likely it would be, in the long run, that regulatory agencies would want to see that data as justification for costs. Take the case of A.T.&T. They ended up not having their books fully examined because the government said they didn't have enough people to examine all the books. If there were a data base system of the kind contemplated by the membership in this room, that data would be machine readable and useable. Apparently it is not. For lack of such information, a democracy suffers.

KAMERMAN:

I would like to state a counter thesis which says, that as we provide more information in a readily available or correlated form through information systems, we are perhaps sounding the death knell of progress in the sciences, destroying management decision making ability, and other dangerous things. There are a couple of reasons why that is not such an outladish statement. Information systems tell us history, and they tell us it very well. The better they tell us history, the more we are likely, I would contend, to do tomorrow what we did yesterday because we know how much that did yesterday, and the less we are able to justify those far-out concepts which do frequently become the watershed of great forward steps in the sciences. Information systems also present to management, at all levels, strict accountability. People tend to be self-protective. When one gets into an area of strict accountability, where it is possible to find out the effects of a decision, I would suggest that two things are likely to occur. One: we move from one area of decision theory to another which is dealing with probability disaster. We choose instead not to take a course which might be dangerous and judged later. Our systems don't tell us what the future will be, but they will make me accountable if I make an improper decision: Two: we may in fact cause, in an informal organization, a general agreement towards mediocrity in all cases because if everyone plays it safe, we are in a very good condition.

BERG:

It seems to me that you are saying accountability, the availability of information, leads to conservatism in science and management because it means, to the extent people can read historical information, they become cautious rather than bold. It seems to me, however, that that is what we have now without the information.

No one with or without a data base management system is going to eliminate

the informal group. What one can do with it is temper and reduce the unproductive consequences and the effects of these informal systems. It is not in the nature of a human being to contemplate or countenance the elimination of the arrangement he makes with other people, because knowledge is always imperfect. One of the things I have stressed, I hope, was that the availability of information will highlight to managers how much of their decision making is ultimately a value judgment. We can at least clear the air and join issues with this kind of information, and I wouldn't see the hazard to the scientist. It seems to me that it is almost an irrational approach to knowledge to contemplate that knowledge is almost by definition a hinderance. There are unimaginative people who develop "psychological sets" and they become more entrenched or rigid as time goes by. Lord knows, we have the research to prove that. I can create the beginning of a psychological set in the audience that will lead them to do asinine things like lose expensive wagers to me. That is relatively easy to do. It is also easy to make people aware how rigidities creep in as a result of psychological sets. But information per se need not be inhibiting. It could be a liberating component in decision making. I would be less restless than you about a little less "ad hocracy" and a little more refinement in what I think of as bureaucracy.

CODD:

The study that you described appears to be a valuable contribution as a piece of sociological experimentation. However, you appear to be drawing some very dangerous conclusions from the experiment. The implication that you appear to be drawing when you talk about egaletarian pay, etc. is that this is a proper measure for the value of that employee. That is, the productivity on the job is a proper total measure of the value of that employee to the company or to the organization. This seems to me a very narrow viewpoint especially in this day.

BERG:

Before you go any further, let us draw a distinction between what I imply and what you infer.

CODD:

Now it seems to me that this conjures up the image of employees being viewed like packages of software: things to be tuned up to the greatest efficiency on a particular job that the company wants done now. Any organization worth its salt is going to put a high value these days on education, because it needs flexibility. I want to add one more point that will tie this in with data base management systems: that it is just as important to employ flexible adaptive people in your company today as it is to employ flexible adaptive data base management systems.

BERG:

In the limited time available, let me say, first of all, I did not intend, or even vaguely wish to imply, that educational screening requirements should be the only ones used by management for any such person. Leaving aside what I infer and what you may not have implied to be a relationship between more schooling and more flexibility (which is an interesting hypothesis unsupported by any data that I know), let us assume that better educated people are in fact more flexible. I would suggest that this is a managerial problem as well as a personal problem. These are settings in which one wants more and in which one wants less flexibility. That raises a different set of selection questions than simply formal education. My concern with education came from a different side.

We have been using education as a screening device during the long period of the so-called loose labour market. The question was whether management has

benefited from that decision. One of the several measures that I took of the cost-benefits involved, is what happens to the better educated employee and what happens to management. What I discovered was that management has great difficulty with the better educated employee. I can demonstrate that the level of job dissatisfaction of a better educated worker who is working with less educated workers exceeds by two standard deviations the discontent one finds between the less educated co-worker in the same job setting. That is not trivial for management.

CODD:

What is the moral of this point?

BERG:

The moral of the point is a very simple one. If management is using a strategy, the cost and benefits of which are not clear, and they have data which makes it possible for me as a scientist to make some assessments of the cost and benefits, I would suggest that one of the costs that you may have overlooked, in reading inferences about my study, is that excess education is problematic to management. One can do a more refined assessment and come to exactly your conclusion: that the firm may in fact need more and better educated people, but that is an empirical question. It is not a question you presume on the basis of deeply felt attitudes about the assumed impacts of education on the individual.

But I have already taken a good deal of your time, and you have been most patient in hearing me out. I hope I have not been too wide of the mark in my speculations. As a student said to me at the end of my very first lecture at Columbia, 15 years ago, as he became agitated listening to something that I was saying, "Professor Berg, if what you say is true, you might be right". If what you say about the impact of education is true, that it breeds more flexible people and has no costs to management, then I would have no arguments with you.

Data Base Management Systems, D. A. Jardine (ed.).

DATA BASE SYSTEMS - IMPLICATIONS FOR COMMERCE AND INDUSTRY

T. B. STEEL, JR.
The Equitable Life Assurance Society of the United States
New York, New York

The objective of this paper is to explore some of the consequences to commerce and industry of the availability of data base systems that are comprehensive, reliable, secure and demonstrably cost-effective. By such systems I do not mean currently available half-way houses such as IDS, IMS and TOTAL, even with the known defects repaired and the obvious enhancements incorporated. Rather, I want to consider the situation when systems are available that satisfy most of the wish lists presented in other papers at this Conference. I will speak to the questions of:

1. what opportunities are created by data base systems availability, and
2. what institutional adjustments are required to permit these opportunities to be exercised.

Prior to explicit consideration of these questions it is necessary to explore briefly the anticipated environment in order to identify critical business problems which will present themselves. There is inevitably a synergistic effect between available technology and applications demands. Compelling business needs drive research and development to manufacture solutions of particular kinds while new technology tends to generate its own uses. This interplay is at the heart of what follows.

There is an inherent time lag between conceptual inventions and commercially available products employing those inventions, whatever the field. Reduction to practice, engineering development, product design and market preparation all take time. Considerable evidence exists to support the thesis that this time lag is seven years in high technology fields such as information processing. Examples include the transistor, holography, procedural programming languages and operating systems. One hears argument that this is no longer true of software concepts but the examples offered are unpersuasive. Thus a technological horizon of 1980-1985 is established for the initial utility of concepts explicated in the next few years.

The environment reviewed must be that extant for the decade beyond the technological horizon. This follows from two facts apparent in contemporary practice:

1. In-place systems have a non-zero life span and, therefore, must have the capacity to satisfy requirements for some period of time - often several years - following initial installation. As the capital investment in information processing systems increases this tendency will become more compelling.
2. Many, if not most, installations prefer not to be on the "leading edge" and continue to satisfy contemporary needs with ten year old technology.

This last point is amply demonstrated by the existence of a real, albeit moderate, used 7080 market.

The point of the discussion above is that, like it or not, the applications systems intended for use in the time period 1975-1985 are being implemented by our installations today, by and large employing yesterday's technology. So long as the seven year rule on the lag between conception and product availability remains valid, much of the real contribution of this Conference will impact the applications systems to be designed and built in 1980-1985 for use in the 1980-1995 time frame. Accordingly, the initial objective of this paper will be an identification of reasonable bounds for the relevant environmental parameters that will condition information processing needs during the period 1980-1995.

The approach taken is to construct a rough scenario for the expected course of events during the indicated time and develop an economic model that is

internally consistent, "surprise free" and exact in presentation. Time does not permit exploration of the variances or construction of less likely scenarios. For the purpose at hand it will make little difference even if the explicated scenario is significantly in error. The general trends will hardly be in dispute and it is these trends that carry the essence of the message.

I am using the term "surprise free projection" in the sense of Kahn and Bruce-Briggs (1972); as an extrapolation into the future that is not surprising to the author. It has many of the characteristics of a naive economic model although it is broader, somewhat more sophisticated and, thereby, rather more useful.

Certain major assumptions guide the structure of the model. They are:

1. No major war.
2. No uncontrollable economic dislocation in industrialized countries.
3. No political revolution in a major country.
4. No epochal social revolution occurs.
5. No epistemological revolution occurs.

The following elaboration should clarify these assumptions and permit the reader to assign his own probabilities to their validity. For a war to count as "major" there must be the real potential for nuclear weapons use. The Arabs and Israelis will likely have another shooting match in the time period of interest but so long as NATO and the Warsaw pact stand aside such a conflict is not to be considered "major".

The kind of economic debacle to which the second point refers is that for which the modern architype is 1929-1939. Certainly no economic event since the commencement of the Marshall Plan would satisfy the failure of the assumptio . Minor dislocations which may be painful to those involved but which do not significantly interrupt the long term trends are both expected and irrelevant.

The third point is obvious with the caveat that "revolution" means what it says. An internal Kremlin power struggle or the constitutional impeachment of the U.S. President would not count.

"Social revolution" is a difficult term to define with any precision. In some sense the entire twentieth century can be viewed as a time of continuing social revolution. If we take the present pace of social change as the norm, then "social revolution" in the current context must mean the consequences of an event that is both extremely rapid and highly dramatic. Martin Luther counts but the contemporary counterculture does not. Given the validity of the other assumptions, this one can probably fail only for a reason that is both bizarre and religious.

The last point requires some explanation. It is perhaps easiest understood in the presence of some examples of phenomena that would cause it to be violated. Thus are ruled out such occurrences as: an irrefutable demonstration of the validity of astrology; proof of the existence of extraterrestrial intelligence; invention of a practical matter duplicator; development of reliable telepathy; and, most important, X.

Comparable historical events, given today's telescoping of time due to modern attitudes toward change and instant communication, are such occurrences as the discovery of the smelting process, the inventions of moveable type and the stream engine, and the successful return of Columbus.

Caution must be urged on this issue. Cause and effect are extremely difficult to disentangle and the seemingly trivial can have enormous consequences. Indeed, it can be persuasively argued that the invention of the horsecollar in the thirteenth century as the death knell of feudalism from which the American, French and Russian revolutions followed, thus giving birth to the modern world. It is sufficient, here to postulate that our current world-view will not be radically altered in the next two decades.

Given these assumptions, which can be summarized as military, economic, political, social and epistemological stability, a model of the next twenty years can be constructed with reasonable precision. A prudent betting man might give four to one odds against the forecast being more than one standard deviation in error. Tables I-II (Appendix) indicate the major parameters of the economic

component of the model. Two points in connection with the displayed numbers should be noted. First, the UN definition of GNP is employed for consistency; therefore the data for the UK and USA do not accord exactly with domestically published numbers. Second, again for consistency, the extrapolations have been smoothed and adjusted and so do not represent exact extractions from the cited sources. As the objective here is to identify major trends and draw inferences from those trends, this is a defensible procedure, conventional econometric practice notwithstanding.

Two major points are apparent at once from the data. First, the world is sharply dichotomized in economic strength between the developed and underdeveloped countries. Twenty percent of the world's population enjoys eighty percent of its wealth and the disparity is getting larger, not smaller. Furthermore, it is only the developed countries that have the surplus resources to invest in high technology enterprises like data base systems. For the rest of this century, at least, the action is going to remain where it has always been: North America, Europe, the USSR and Japan. So much is almost middle school social studies with its emphasis on "have" and "have-not" nations.

The second point is less well known but equally valid and more pertinent to the present issues. Already true in the United States and becoming true across the board in the developed countries during the time span considered in entry into the socio-economic bracket increasingly called "post-industrial society". The precise definitions vary but a rough indicator is a per capita GNP in excess of $4000-$4500 (US 1970). Not all experts are agreed on whether this is truly meaningful or whether it has really occurred yet in the United States, but most do agree that significant changes are in process (Bell (1973)). Its impact on the subject at hand is profound in any event.

The principal characteristics of post-industrial society are rather neatly summarized by Kahn and Wiener (1967) from whom the following is derived:

. Economic activities are service-oriented rather than production oriented
. Continued urbanization and trend to megalopolis
. Income "floor" established
. Efficiency no longer primary
. Widespread "cybernation"
. "National interest" values eroded
. "Puritan ethic" eroded
. Knowledge industry paramount
. Culture is sensate, secular and humanist

All of these observations bear to some degree on the nature, diversity and acceptance of future data base systems. Indeed, it is the major thesis of this paper that these coming socio-economic changes would force the invention of data base systems if they were not already conceived, and that the pressure to use, expand and improve such systems will be enormous and irresistable, despite the known and feared undesirable side effects of the widespread installation of systems providing rapid and complete access to stored data. In this observation I am disagreeing slightly with Westin and Baker (1972) in their evaluation of the danger inherent in the present situation and adding weight to their thesis that now is the time to take precautions against future misuse. As this paper is concerned principally with the commercial and industrial impact of data base systems, it is not the proper vehicle for a dissertation on privacy and freedom excepting only to observe that if the conclusions of this paper are valid, these questions have even more importance than has yet been given them.

Examining Tables 9-11 (Appendix) in some detail, one can appreciate the possibilities of attaining the indicated productivity increases in the traditional primary and secondary industries through devices such as conventional automation. The problem of realization comes in the tertiary and quartenary industries (transportation, trade, finance, services and government) and the associated management and marketing functions of the other types of industries. Conventional automation has its impact here but hardly enough to provide the indicated 250 percent improvement. This can only come, so far as any technology extant suggests, from organization, concentration and effective resource management which means

information availability and, thus, data base systems.

On a larger scale, the indicated principle of erosion of "national interest" values is concomittent with the continued growth of multinational in corporations. More and more of the economic activity of the world is being conducted by enterprises that are multinational in scope and outlook. Perlmutter (1971) notes that by the 1980s "a unitary global industrial-commercial system, a worldwide network of industrial, financial and commercial activities...could produce 50-75 percent of the gross national product of the world. The key participants are likely to be 200 to 400 supergiant corporations doing a billion to 160 billion dollars of annual sales".

Pressure exists from several directions to elaborate and extend various kinds of government services. The existence of some form of income floor, universal medical care (whether supplied, insured or simply paid for by the government), natural resource management and vastly expanded educational facilities are all but inevitable. All of this requires the availability of a wide variety of information in an easily accessable form.

In order to manage the increased flow of money engendered by the general economic growth, especially with the increased international integration of economics that is indicated, a major restructuring of financial institutions is going to occur. The volume of paper currently being moved around the world is already absurd and the impending growth is impossible. The Monetary and Payments System Planning Committee of the American Bankers Association (1971) is already moving toward a series of recommended actions that will, in effect, make the entire banking system records activity in North America one big data base. The credit records assimilation activities, as reported in Westin and Baker (1972), go hand in hand with this bank activity.

These major trends will be supplemented by many other, only relatively less significant activities implying data base systems. The GUIDE/IBM Delphi Study (Wylie (1971)) lists many such for the indicated time frame; indeed, far too many to list. Perhaps the most significant concensus in this study is the expectation of the price-performance measure of mass storage to improve by three orders of magnitude by 1985. Much of the remainder of the study is an elaboration of the implications of that expectation.

Many scoff at the grand schemes suggested by these various studies, often on the grounds that there is insufficient money to pay for all the ideas or else that the data processing industry is maturing and reaching a plateau. In a world where large corporations can seriously analyze the cost-benefit trade-off of a private corporation communication satellite, the first is a curious reaction. As to the second, it should be recalled that IBM believed in 1955 that all the world's computing needs would be satisfied with 30-40 model 704's (ignoring the AEC and USAF who were both viewed as a bit crazy). As to the required investment, Japan has a proposed program to accomplish much of what has been suggested and, in some directions more, at a cumulative cost of 20 trillion yen ($65 billion a la 1970). This program (Japan Computer Usage Development Institute (1972)) calls for less annual outlay that NASA received through the APOLLO years.

Putting together all the available data and considering data base systems only, it is reasonable to expect in the late 1980's perhaps 250,000 distinct data base systems (distinct in the sense that they are independently useful and used) containing 5.10^{14} bytes (exclusing a few library systems each containing on the order of 10^{15} bytes). Perhaps 25 percent of these data base systems will be physically connectable with at least some others. A reasonable estimate of the investment required for their establishment is $100 billion. As was argued earlier in the paper, the groundwork for these systems is being laid right now. It behooves us all to keep this in mind and not to get too concerned over the fourth bit from the left.

The remainder of this paper is concerned with the necessary steps to take at the moment to minimize the coming pain and suffering that will be associated with the indicated developments. Five major points are worth consideration.

1. Management awareness
2. Vendor effectivity
3. Research utilization

4. Academic responsiveness
5. Standards

First, managements - all the way to the top - must become aware of the importance of data base systems and their consequences. The game is "you bet your company" and it is the only game in town. Corporate managements must become far more aware than they are now, in general, that data base systems are going to become a central element in the management control and operations of almost every business. Further, it is incumbent upon management to develop organizational structures that will permit effective utilization of the necessary tools.

The "Data Base Administrator" concept, so much talked about in recent months, will of necessity become a reality. It will take a number of years to obtain sufficient experience with this new role to permit objective assessments of various alternate modes of providing this function and such assessments are essential to proper functional design of systems.

Managements have an annoying habit of asking pertinent questions such as "how much will that cost?" At the moment there do not exist sufficient satisfactory tools to generate responsible answers in many cases. Definition of such tools (measuring and monitoring devices, system models, data reduction and analysis procedures, reporting techniques) has not yet been adequately accomplished. Furthermore, there is a design feedback between tool definition and identification of the kind of questions that should be asked. Again this calls for management education.

Another issue that management must face is that of security. The appropriate posture for sound physical security is rather well understood if not always practiced. Fire, flood, power loss, mad bombers, disgruntled employees and simple frauds can, by and large, be held in check by conventional protection, prevention and recovery mechanisms. Data base systems, however, involve another dimension of security - information protection.

This subject is one about which much is known and very little published, primarily because its principal area of application has been government communications security and almost everything pertinent is classified "Top Secret - Burn Before Reading." Kahn (1967) provides a provocative glimpse behind this shroud of secrecy and indicates some hint of the relevant considerations.

The need for information security is derived from two sources: privacy considerations and commercial espionage prevention. While the motivation, ethical considerations and regulatory legislation will be radically different on these points, the basic principles employed to achieve the necessary security are essentially the same. Encryption has been the preferred solution from the time of Julius Ceaser to the present. Feistel (1973) discusses some current techniques applicable to data processing.

Two principles of information security need to be better understood. First, there is no such thing as an "unbreakable code" in practice. There are theoretically unbreakable systems but they must be used by people and people make mistakes which can lead to loss of security. The second principle derives in part from the first. You can get whatever degree of protection you want below perfection if you are willing to pay for it. Thus, the key to satisfactory information security is to determine the value of the information to the potential unauthorized user and then design a security system that will make it more costly than that value to obtain without permission. The key here is that it must be the expectations of the thief that guides the expenditure of security resources, not the intrinsic value of the information to the possessor.

Management is going to need understanding of these principles because it will no longer be the case that the principal tasks of the security officer are to see that the fire drills occur and that the guards have polished buttons on their uniforms. A total enterprise approach is necessary to provide true security as the possibilities for compromise are widespread and often subtle. Further, insuring any real degree of security can be expensive. Without explicit examples to dissect it is not possible to provide hard numbers, but data bases with really sensitive information content could require as much as 30 percent extra cost to provide appropriate protection.

The subject of vendor effectivity has been discussed almost ad nausean in every forum on data base systems and need not be elaborated in detail here. It is sufficient to observe that with the widespread growth of data base systems and the obvious integration into the mainstream of corporate activity, it simply will not be acceptable to have systems that are as error ridden as are those presently being purveyed. This applied whether the "vendor" is a supplier in the public marketplace or an internal corporate department. Several orders of magnitude improvement in the robustness of systems is becoming imperative.

A problem that plagues data processing generally is the lack of coupling between the research community and the practical world of application. A great deal of research is done in our field but much of this work, including a lot that shouldn't gets published and subsequently ignored. (The double entendre is deliberate; much gets published that shouldn't as well as much getting ignored that shouldn't. Possibly the former is partially responsible for the latter.) In a sense this observation is a complaint that programming is still a craft or, to use T. Dolotta's happy phrase, a "cottage industry". An engineering discipline is needed, based on the computer science equivalent of basic physics.

This situation is the fault of nearly everybody. Too many partially qualified individuals are accepted for graduate study in data processing related fields; a large number of poorly done theses are accepted; the resultant input to faculties is consequently of uneven quality; the high calibre work that is produced is either misunderstood or ignored by industry; and industry generally criticizes in a carping, rather than constructive manner.

If the whole structure suggested earlier in this paper is not to collapse of its own weight, something will have to be done to rationalize the relationship between academia and industry.

Related to the above point is the issue of academic responsiveness in the curricula. The typical computer science graduate may well be highly trained to write clever compilers for both real and exotic languages and, perhaps, to prove theorems about the correctness of four line ALGOL 68 programs. Often the result appears to industry as analogous to someone who has prepared for a surgical career by concentrating on a study of molecular biochemistry. In this area it is largely the universities and the data processing professional societies that are at fault. Too much attention is paid to academic respectability and insufficient to the real need for the equivalent of an engineering discipline. The care and feeding of 120,000 data base systems is going to force a change; it remains to be seen whether it is accepted gracefully or swallowed painfully.

Finally, the question of the need for industrial standards in several areas is already becoming significant. Both national and international standards are needed in areas such as data elements, communications formats, interface languages and the like. Studies already underway may well suggest other needed areas of standardization. It is going to be vital to successful implementation of large, integrated, distributed data base systems to have such standards well thought out in advance of systems design. Industry generally will be defaulting on its obligations and failing to look after its own needs if it defers without thought to the government and vendors in the development of standards.

A rough rule of thumb suggests a 20 percent additional cost for systems of considerable complexity in the absence of standards. While even without an organized program there will be some de facto standards, the amount of wasted resources would be staggering; perhaps $10 billion (half of the 20 percent).

In summary, the expected direction of the world's economy is going to generate enormous demand for data base systems as an integral part of managing and operating that economy. Further, the nature of the expected growth is dependent on the development of an effective data base technology; so much so that it will force this development almost without regard to cost. In order to maintain control over what can become a runaway locomotive, there are a number of imperatives including high level management awareness, substantially improved system quality, increased coupling between industry and the educational system, and a sound program of standards development.

APPENDIX

ECONOMIC PARAMETERS 1970-1995

TABLE 1
Population - Major Countries[1]

	1970	1975	1980	1985	1990	1995
China	826	903	975	1052	1122	1197
India	548	616	685	761	835	909
USSR	247	260	278	296	315	334
USA	206	215	231	245	259	273
Japan	102	107	111	116	119	123
West Germany[2]	60	61	62	64	65	66
UK	56	57	58	58	59	59
Italy	54	55	56	57	59	61
France	52	54	56	58	60	62
Canada	22	24	26	28	30	33

1 - Millions
2 - Including West Berlin
Sources: China - "Indicators of East-West Economic Strength", US Department of State, 1965.
USA - US Bureau of Census, 1971.
Others- UN Demographic Yearbook, 1965.

TABLE 2
Population - World Regions[1]

	1970	1975	1980	1985	1990	1995
Developed Countries						
Western Europe	353	362	371	379	387	394
USSR	247	260	278	296	315	334
USA & Canada	228	239	257	273	289	306
Eastern Europe	104	108	112	115	119	122
Japan	102	107	111	116	119	123
	1034	1076	1129	1179	1229	1279
Underdeveloped Countries						
Asia[2]	1100	1245	1410	1580	1740	1930
China	826	903	975	1052	1122	1197
Latin America	271	313	362	417	474	533
Black Africa	238	267	305	346	399	457
Middle East & North Africa	163	187	216	248	282	318
	2598	2915	3268	3643	4017	4435
Other	62	72	80	89	100	113
World Total	3694	4063	4477	4911	5346	5827

1 - Millions
2 - Excluding China and Japan
Source: UN Monthly Bulletin of Statistics

TABLE 3
Gross National Product - Major Countries[1]

	1970	1975	1980	1985	1990	1995
USA[2]	1020	1270	1570	1970	2460	3060
USSR	448	572	730	934	1190	1520
West Germany[3]	166	207	258	321	400	498
Japan	143	204	294	376	480	613
UK[2]	142	178	209	256	310	378
France	138	178	216	269	334	416
China	123	178	242	338	475	665
Italy	83.6	104	130	162	202	252
Canada	72.9	93.1	118	151	193	247
India	72.9	93.1	118	151	193	247

1 - 1970 US Dollars in billions
2 - UN definition of GNP
3 - Includes West Berlin
Sources: USSR & China - "Indicators of East-West Economic Strength", US Department of State, 1965.
Others - UN Monthly Bulletin of Statistics.

TABLE 4
Gross National Product - World Regions[1]

	1970	1975	1980	1985	1990	1995
Developed Countries						
USA & Canada	1093	1363	1688	2121	2653	3307
Western Europe	745	925	1130	1400	1720	2140
USSR	448	572	730	934	1190	1520
Eastern Europe	167	214	274	349	446	570
Japan	143	204	294	376	480	613
	2596	3278	4116	5180	6489	8150
Underdeveloped Countries						
Asia[2]	162	204	252	311	381	476
Latin America	127	155	193	239	299	374
China	123	178	242	338	475	665
Middle East & North Africa	52	67	86	110	141	179
Black Africa	30	37	46	57	70	87
	494	641	819	1055	1366	1781
Other	64	77	96	119	149	185
World Total	3154	3996	5031	6454	8004	10116

1 - 1970 US Dollars in billions
2 - Excluding China and Japan
Source: UN Monthly Bulletin of Statistics.

TABLE 5
GNP Per Capita - Major Countries[1]

	1970	1975	1980	1985	1990	1995
USA	4951	5920	6800	8040	9500	11200
Canada	3310	3880	4540	5400	6440	7480
West Germany[2]	2770	3395	4170	5020	6160	7550
France	2660	3300	3860	4640	5570	6710
UK	2520	3120	3600	4410	5260	6400
USSR	1815	2195	2620	3150	3780	4550
Italy	1547	1890	2320	2840	3420	4150
Japan	1400	1905	2650	3240	4030	4990
China	149	197	248	321	424	555
India	133	151	172	198	231	272

1 - 1970 US Dollars
2 - Including West Berlin

TABLE 6
GNP Per Capita - World Regions[1]

	1970	1975	1980	1985	1990	1995
Developed Countries						
USA & Canada	4800	5700	6570	7780	9190	10800
Western Europe	2110	2550	3050	3700	4450	6410
USSR	1815	2195	2620	3150	3780	4550
Eastern Europe	1605	1980	2450	3040	3750	4670
Japan	1400	1905	2650	3240	4030	4990
Net	2550	3040	3690	4400	5280	6370
Underdeveloped Countries						
Latin America	470	495	533	574	632	703
Middle East & North Africa	318	358	398	444	500	563
China	149	197	248	321	424	555
Asia[2]	147	164	179	197	218	246
Black Africa	126	138	151	165	176	190
Net	190	220	250	289	340	402
Other	1030	1070	1200	1340	1490	1640
World Net	853	975	1120	1310	1500	1740

1 - 1970 US Dollars
2 - Excluding China and Japan

TABLE 7
USA Population - Distribution by Age[1]

	1970	1975	1980	1985	1990	1995
4	17.5	21.5	24.2	25.7	25.8	26.3
5 - 9	20.3	17.3	21.3	24.0	25.4	25.5
10 - 14	21.1	20.3	17.3	21.3	24.0	25.4
15 - 19	19.4	31.0	20.2	17.2	21.2	23.9
20 - 24	16.6	19.3	20.9	20.1	17.1	21.1
25 - 29	13.6	16.5	19.2	20.8	20.0	17.0
30 - 34	11.5	13.5	16.4	19.1	20.7	19.9
35 - 39	11.2	11.4	13.4	16.3	19.0	20.6
40 - 44	12.1	11.1	11.3	13.2	16.1	18.8
45 - 49	12.2	11.9	10.9	11.1	13.0	15.8
50 - 54	11.2	11.9	11.6	10.6	10.8	12.7
55 - 59	10.1	10.8	11.5	11.2	10.2	10.4
60 - 64	8.7	9.5	10.2	10.8	10.5	9.6
65 - 69	7.1	7.8	8.5	9.2	9.7	9.4
70 - 74	5.6	5.6	6.1	6.7	7.2	7.6
75 - 79	3.8	3.9	3.9	4.3	4.7	5.0
80 - 84	2.3	2.3	2.3	2.3	2.5	2.8
85	1.5	1.4	1.4	1.4	1.4	1.5
	206	215	231	245	259	273

1 - Millions
Source: US Bureau of Census, 1971

TABLE 8
USA Labor Force - Distribution By Age & Sex[1]

	1970	1975	1980	1985	1990	1995
Male						
19	5.0	5.5	5.3	4.4	5.5	6.2
20 - 24	7.0	8.2	8.9	8.5	7.2	9.0
25 - 34	11.8	14.1	16.7	18.7	19.1	17.4
35 - 44	10.9	10.7	11.7	14.0	16.6	18.6
45 - 54	10.5	10.8	10.2	9.9	10.8	13.0
55 - 64	7.1	8.0	8.5	8.6	8.1	7.8
65	2.2	2.0	2.0	2.1	2.1	2.2
	54.5	59.3	63.3	66.2	69.4	74.2
Female						
19	3.3	3.6	3.5	3.0	3.6	4.1
20 - 24	4.2	5.1	5.6	5.5	4.7	5.8
25 - 34	4.9	6.0	7.3	8.3	8.5	7.8
35 - 44	5.5	5.7	6.3	7.6	9.2	10.6
45 - 54	6.6	7.2	6.9	6.9	7.6	9.1
55 - 64	4.2	4.8	5.3	5.5	5.2	5.1
65	1.1	1.2	1.3	1.4	1.5	1.5
	29.9	33.6	36.2	38.2	40.3	44.0
Total	84.4	92.9	99.5	104.4	109.7	118.2

1 - Millions
Source: "Labor Force Projections for 1970-1980: Special Labor Force Report No. 49", US Bureau of Labor Statistics, 1965.

TABLE 9
USA Labor Effectivity

	1970	1975	1980	1985	1990	1995
Labor Force[1]	84.4	92.9	99.5	104.4	109.7	118.2
Employed[1]	81.2	88.7	95.0	100.0	105.3	113.5
Unemployed[1]	3.2	4.2	4.5	4.4	4.4	4.7
Mean Work Week[2]	38.3	37.8	37.2	36.7	36.0	35.2
Hours Worked[3]	3.10	3.35	3.54	3.67	3.80	4.00
Output per Man-hour[4]	1.00	1.15	1.35	1.63	1.97	2.32
Total Output[4]	1.00	1.24	1.54	1.93	2.41	3.00
GNP per Man-Year[5]	12.6	14.3	16.5	19.7	23.3	27.0

1 - Millions
2 - Hours
3 - Billions per Week
4 - Index: 1970=1.00
5 - 1970 Dollars in Thousands
Source: "Statistical Abstract of the United States", Department of Commerce, 1970.

TABLE 10
USA Labor Distribution by Industry[1]

	1970	1975	1980	1985	1990	1995
Agriculture	3.0	2.9	2.4	2.0	1.8	1.7
Mining	0.6	0.6	0.5	0.5	0.5	0.5
Manufacturing	18.9	19.8	20.2	20.5	20.8	21.0
Construction	3.6	3.9	4.0	4.7	5.0	5.3
Transportation & Utilities	4.3	4.6	4.8	4.9	5.3	5.6
Trade	17.5	19.8	22.4	23.6	25.0	27.8
Finance & Real Estate	3.4	3.8	4.2	4.5	5.0	5.5
Services	15.6	17.5	20.0	21.4	22.1	24.7
Government	14.3	15.4	16.5	17.9	19.8	21.4
	81.2	88.3	95.0	100.0	105.3	113.5
Government						
Military	3.1	3.0	2.9	3.0	3.1	3.1
Civilian Federal	2.8	2.8	2.9	3.0	3.2	3.3
State & Local[2]	4.3	4.7	5.9	6.7	7.9	9.0
Education	4.1	4.5	4.8	5.2	5.6	6.0

1 - Millions
2 - Excepting Public Education
Source: "Statistical Abstract of the United States", Department of Commerce, 1966.

TABLE 11
USA Gross National Product - Major Components[1]

	1970	1975	1980	1985	1990	1995
Personal Consumption						
Durables	99	130	170	222	290	379
Non-durables	279	328	386	455	535	630
Services	269	347	439	583	735	931
Total	647	805	995	1250	1560	1940
Domestic Investment						
Residential structures	37	44	53	63	77	92
Non-residential structures	38	48	60	77	96	120
Producer durables	76	96	120	151	191	241
Change in inventory	8	10	12	15	19	24
Total	121	150	185	229	287	357
Government Purchases						
Federal	84	97	110	132	156	182
State and Local	120	157	204	262	336	430
Total	204	254	314	394	492	612
Net Exports	10	13	16	20	25	31
GNP	1020	1270	1570	1970	2460	3060

1 - 1970 US Dollars in billions
Source: US Department of Commerce.

DISCUSSION

JARDINE:

I would like to express my support for Tom's comments about the university's role. Somebody very accurately described a computer scientist as a person who in two months can write a compiler for a language that nobody wants or needs. The problem has also been correctly identified, but I don't think it's as one sided as Tom put it. It is certainly the case that the universities have not been given any sense of direction by the computing industry as to their requirements, but the universities have not solicited that support. There is another problem that has not been addressed by the universities and that is exactly what Professor Berg talked about, the clique, the closed society of the university which makes it extremely difficult and sometimes very uncomfortable to bring in a person from the industrial community. Tom has made a very good analogy that if we are going to turn out competent people at this stage in the development of university trained graduates we should be adopting more and more the philosophy of the engineering schools, who have managed to keep a close contact with the industries that they serve, and who have had an extensive cross fertilization with those industries. They are not nearly as hide-bound in their selection of people as the formal disciplines.

STEEL:

If I may just add one point to that, I think you did not mention (although I don't think you meant to deny it) that the engineering departments have maintained an appropriate level of contact with the physics and mathematics disciplines. We have to do that as well, we can't simply slice off the theoretical side.

MAYNARD:

In one of your last points on standards, you made a statement that management doesn't understand spending some money, a few dollars, to save 10% of a certain amount of money. That is upsetting but what can we do about it?

STEEL:

That is not really quite what I meant. Obviously managers understand the principle of investing a little money to save a lot. The point is we don't seem to know how to explain that you can save that kind of money, with a relatively modest investment. The petroleum industry will spend millions sending people to API standards activity. General Motors participates very heavily in the standards activities concerned with automobiles. My industry is very concerned with standards relating to paper pushing and money pushing. When it comes to standardization of a tool that is used by the business, it is very difficult to convince management that the user of the equipment has a real stake in standards. Data processing is peculiar in this respect. We build half of the product we use. It isn't delivered in a little box from a manufacturer. Virtually everybody in this room would be out of a job if it were that simple. The point is, everybody is in the systems manufacturing business, not just the vendors. That is not adequately understood by our management.

HEINONEN:

The concern about performance ^specially as it relates to recovery and roll back were something that I think the users would want the vendors to have a clear concern about as they left this session. I don't think enough emphasis was put on the need for a more sturdy data base - data communications system.

STEEL:

When you spoke about performance, I do not want the manufacturers to walk away from here thinking that we mean performance in the nitty-gritty sense. We mean total systems performance, not sub-optimization.

SIBLEY:

I would like to speak as an engineer, professor, user, executor, and victim, first of all as a victim. I'm a victim of university professors as well as I'm a professor in a university. I'm also a victim of the real world situation which is that we're dealing with systems bigger than we have ever seen in our life before. Consequently there is no technology to deal with these massive systems; there is no knowledge on our part as to how these are going to affect the system into which they are introduced. We're living in essentially the era of trying to measure (a) where we don't know what we are trying to measure, and (b) we don't know what the effect is of putting the measure in. We just have a gut feeling that we're going to make things better. Now the trouble is that we have no place in which to do much of our research. The people really concerned have essentially no lab. They can't do it in a small scale because if you try to, you reach utterly incorrect and fallacious conclusions, I feel. Equally, industry is not in a viable state whereby they can invite someone to experiment with a running system. We have previously broken down big systems into small systems. Unfortunately, here we see overlapping systems. We have different operations, we have different production, we have different management and we really have no methodology for management. We are also putting on top of this social systems which we are going to try to opeate with, and again we have no technology.

STEEL:

Your comment on the lack of laboratories is a very good one, and that's where a great deal of this 65 billion dollars the Japanese are proposing to spend will be used. We ought to be doing the same thing.

HOFFMAN:

I'm concerned about one of the underlying assumptions which is that society is going to accept the direction of the projections you had on the slides. As time goes by and as the service phase of the economy increases we have more and more people who actually do nothing. We'll build these massive data base systems with a large number of people operating, developing, maintaining, extending, documenting, and standardizing them. We may end up with a situation which is generally called the quality of work issue. Are people really going to be motivated to produce high quality systems? I think we might end up with consuming such a proportion of the world's resources and producing so little, we will be in a very immoral position.

KAMERMAN:

I would like to draw an analogy between standards and law. Law is a structuring of the society. It lays out the rules, the regulations, the way things are done, and has been rather successful over the years in doing that job for most societies, certainly for North American societies. One of the reasons it has been successful does not exist in the computing standards community. That is, once one has written these laws (which are as equally as bad as any standards: inaccurate, ambiguous, not dealing with the particular problem) there is an entire area of jurisprudence which deals with the continuing interpretation and reinterpretation of law. If we're going to set up standards as a direction we have to go we have to also set up the jurisprudence of standards to allow certain flexibilities, to let us go ahead.

STEEL:

In the law one presumably knows what the penalty is for violating the law. We don't know what it costs us to not have the standards. That was the point I was trying to get at. We damn well better understand what it is costing us not to have it.

TURNER:

Your comments on the products that we get from the vendors are very valid. Two areas particularly concern me. One of these is the state in which the products are in general turned over to the user. I find it absolutely disgraceful that a user has to do the final stages of testing and that the product delivered to the user is in effect unusable. I find it ridiculous that much of our staff is doing work that the vendors should do. The types of testing and quality control that manufacturing concerns have been doing for years virtually does not exist. The computer manufacturers don't use their own product. They send that out to the user domain and expect the users to do that work.

The second point has to do with marketing and the fact that there are a large number of false claims, particularly in the area of data base systems. For example: if you install this particular product, the result will be radically improved economics within the enterprise. There are many problems which are often far more deeply set than just efficiency in computing. These are never addressed. Very broad based claims are made and the data processing professionals are left with a management that believes that there are going to be large scale gains if a particular course is followed. I feel that is a practice which we have to resist.

BERG:

Tom, I don't think you should construe any of my earlier remarks today as undermining the point you were making on the board with connection with the growing meritocracy. The study that I referred to in my own remarks really go to a slightly different point. The subtitle of the study, "The Great Training Robbery" was meant to imply the idea that we need alternatives to formal higher education and the process by which we prepare a young population and an aging population that wishes to retool itself, for new occupational challenges that come before them. As we look at the occupational structure we can see several implications of the trends that you're outlining, straightforward manpower projections. A service economy is different from a mass production automobile economy. The data base systems and the kinds of work you people can contribute will undoubtedly help us get a better fix on the measurement problem. As we understand a great deal more about what people do at work, we will discover that we need a much more differentiated training apparatus than the post war emphasis on straightforward, simple, formal, higher education. For all its many contributions to the culture and the society, it may not be the simple sovereign universal solution to a growing meritocracy. In the short run we have an historical problem of inequality that bumps into the meritocracy issue. It is distinctly self serving, particularly for white males, to see the salaries that they are earning and the positions they hold as reflecting only their merit. It also reflects the booms, the benefits, the pensions, the good jobs, the decent titles, the professional licenses we have. They also are borrowed from a fairly restricted and truncated selection system which may not have given us the meritorious work force that we would like to have. I could see positive trends in that arrangement: the fact that applications are slipping in higher education, the fact that the drop out rates are beginning to be high in the absence of a draft system that compels people to stay in school. In the year 2000 it strikes me as very probable that we will do a much better job of measuring what people do, what is required of them. That's not going to hurt, that will help, higher education because it redefines higher education's mission and in many respects simplifies the judgments we have to make in respect to the questions of supporting higher education.

UHRBACH:

I would like to address an earlier point, that data base processing management advises its own management. One of the problems that we face is an incestuous relation between the vendor and the data processing community which leads, for any number of internal political reasons, to larger and larger configurations, increased cost, and proliferation of staff. Maybe we should be rethinking the way in which we organize our systems, from the standpoint not of whether we can do a better job with more equipment, but whether we can do a better job, using some data base tools available today, with less equipment, perhaps in a different mix.

STEEL:

I think that is beginning to change. I see signs in a number of companies that they're beginning to understand, at fairly senior management levels, that they have been put on for 20 years. They're starting to put their thumb down. The message is also beginning to get across that 50 of those guys are better than 400 of those. That's another way of phrasing your point: take high quality work and do it right. There are serious economic pressures that are going to force us in that direction.

HOBBS:

I think that the user gets the kind of systems that he deserves, the kind of

systems that he wants. If someone comes up with something that is significantly new and different you won't buy it. I cite again the case of the symbol computer which was a fairly revolutionary development in computer organization and particularly with respect to its impact on software. It got nowhere because it was too different.

STEEL:

We are put in a position where we can't afford to use something new, by what the vendor does to us. He locks us in. We don't have that option and if we had nice clean systems to start with and good standards we would have that option.

HOBBS:

I will again defend the manufacturer. You permit yourself to be locked in. If you insisted, for example, on machine independent compilers you would go a step toward removing that degree of being locked in.

STEEL:

That's where I come to this management awareness problem. This thing has to be escalated high enough up in our companies to get the appropriate kind of pressure on the vendors to give us that. You know people in this room have been yelling about this for 20 years but we don't have enough impact. It is the guy that signs the bill that must refuse to pay or refuse to buy.

Data Base Management Systems, D. A. Jardine (ed.).

VENDOR RESPONSES

EDITOR'S NOTE:

At the end of the conference the representatives of the various hardware and software vendors present were asked to respond to and comment on the discussions both formal and informal held during the conference.

The ground rules under which these responses were made were:

1. no vendor production discussions should be presented;
2. no marketing strategies or issues should be discussed;
3. the responses should be related to the requirements presented by users during the conference;
4. the responses should be limited to fifteen minutes per representative.

Vendor responses follow in alphabetical order of company name.

S. Bedekar (Burroughs Corporation)
T. Richley (CINCOM Systems Incorporated)
R. M. Olson (Control Data Corporation)
H. C. Lefkovits (Honeywell Information Systems Inc.)
H. C. Reinstein (IBM Corporation)
E. D. Scott (National Cash Register Company)
P. A. Hartley (Sperry-UNIVAC)
J. B. Pierce (Xerox Corporation)

BEDEKAR:

I will answer your questions in three basic categories. I will characterize what we consider our system's fundamental characteristics are, then I will give you a brief overview of our solutions to the problem in terms of data management, and finally I will give you some insight into what we believe is the method to solve this problem over the long haul. Burroughs manufactures computer systems known as the B700 through the B7700. These systems are characterized by being programmed exclusively in high level languages to the exclusion of assembly-level languages. We believe that our systems philosophy has been to build language processors, the best example of which is the 1700 which changes its basic instruction set to facilitate the language that it is processing.

In general our systems are multi-processing systems and organized symmetrically with automatic re-entrancy, virtual memory, and virtual machines as the fundamental concept. We have decided to partition the data management problem into two categories. The first category is what we call programming aids. Programming aids usually imply the facility on the part of a set of programmers to do data communication handling, implementing user oriented languages, implementing reporting and inquiry systems and file handling systems. We have used either pre-processors or generators to build these kinds of systems.

The second category of systems that we build is what we call integrated systems. Our belief is that you cannot build a system without consideration of both the hardware and software system taken together. Our data management systems as characterized by the 6700, are a combination of the entire software system and hardware system that is implemented. This includes all languages, all language facilities, all operating system functions and hardware features such as queuers, graceful degradation, automatic reconfiguration facilities, manual reconfiguration facilities, and automatic duplicators. An integrated data management system is not a language; it's not a language interface; it's not an operating system or a set of hardware devices. It's a combination of all of them taken together. It is our belief that specification of user requirement in any one isolated area only solves part of the problem.

We have applied the same principles that have been enumerated for data definition, to data communications. We supply what we can network definition languages which are a logical description of the data communication network that interface to the conventional languages through normal file handling facilities. Our reporting and inquiry systems fall into two categories. The coding systems like the Codasyl report generators and so on, are part of the compilers. In addition to that, conventionally described reports are produced by a user oriented language, by simple description of the data to be reported on the sorting patterns, and so on. To us, an inquiry or interactive system is a normal extension of a desk calculator for the man who has no concern about dealing with a large data base. We can start by just asking the system to add 2 and 2 together and then proceed from there to do data retrieval.

Computer programming and computer design at this point in time is not a science, and for all of us to assume that computers are infallible, because they give reasonable answers in most cases, is a mistake. I realize that this is an extreme point. Once you back off from that, then the reliability-related issues fall into a variety of categories. These are implemented through hardware devices, by instruction re-try and so on, and through the tools that one uses to program these systems.

It is our experience that systems programmed in low level languages tend to be less reliable than systems programmed in high-level languages. That is one facet of the reliability issue, of operating systems, or user programs. A program is essentially the best guess at the solution which you really want to achieve and debugging is essentially the art of proving that there are bugs. Debugging does not prove the absence of bugs. So until some fundamental research is done in this area to contain this problem of programming as an art in a reasonable, commercially practical way, our attempt is going to be to use some of the principles that people like Dijkstra or Hope have illustrated, generally categorized as the structured programming technique.

We believe that this is going to be our first attempt to contain this problem. We believe that the distributed network problem is a matter of both distributed data bases as well as distributed computing power and to that end it is our belief that fundamental work needs to be done in this area to structure systems in such a way that they can distribute work properly and not arbitrarily, which is one of the questions which has been brought up before us.

The last point I'd like to make is that to us the trade-offs as to whether you use a programming aid, an integrated machine, or a large scale or small scale machine, is not necessarily a problem of the user himself. It is a problem that we the vendor need to address. And to that end, the trade-offs as to when we would recommend a programming aid or when we would recommend an integrated system will primarily depend on the cost effectiveness of that system in the user's environment. A case in point would be that we believe that programming aids probably require less sophistication and less administrative control on the part of the user for smaller systems than would an integrated system of the scale of a 6700, 7700, or a 4700.

In conclusion, I would like to reiterate our belief that the solution to the data management problem is not going to come from any one facet of the overall system design. An overall system which is a combination of hardware and software needs to be put together to solve the problem of data management and to that end our first attempt at that has been the 6700 Data Management System.

STEEL:

Could you comment on the Burroughs' view of the Codasyl situation?

BEDEKAR:

Yes, the response that I'm going to give you is a combination of our stated policy and my feelings as an implementor of that system. I'll isolate those two so we know which is which.

Our current vote on the Codasyl proposal is that we voted against the DDLC and the DBTG report. Let me state our reason for this. Let us take the issue of the data definition language and COBOL. As I mentioned earlier, it is our firm belief that a proper notation on the problem is at least three-quarters of the solution. Ad hoc notations are not adequate. To this end we feel that the excessive bias of the data definition language towards COBOL does not serve the interest of other people - such as FORTRAN users, PL/I users, ANSI COBOL users, APL and BASIC users, and we feel that a bias toward a given language is not necessarily a worthwhile thing to have. The second feeling we have is that there is a distinction made in the DDLC as a technical issue between program and data, and we believe that the static description of data is not necessarily a viable description of data. One of the classic examples in traditional forms is sorts. Sort is an algorithm that sequentially produces ordered data. The other aspect of the DDLC to which we object is that it addresses itself only to numeric and alpha data. There are forms of data that are not numeric or alpha. We feel that a proper notation for data definition language that considers all of these language problems is of paramount importance, because of the fact that finally what your vocabulary - and I don't mean in terms of a computer jargon - will allow you to say is all the problems that you can ever solve. In terms of COBOL, we feel that some of the other work that has been going on does not address itself to the area of data management approach. The two specific instances of that is that other language implications have not been considered. What happens if you bind the FORTRAN program to COBOL? What kinds of parameters can be passed? Can you pass a set as a parameter?

The second point we have reservations about is that the CODASYL ATG group has stated standards about asynchronous processing which includes multi-tasking, subroutines, co-routines, events, soft-interrupts, and so on. These have not been addressed and in addition to these two items, the recovery related issues and transaction-processing related issues have some semantic clarity required that we feel uneasy about.

RICHLEY:

Cincom found the conference extremely informative. It was an excellent opportunity for the exchange of ideas and information between the leading users of and developers of data base management systems.

We learned that there are significant differences of opinion as to the desirable features of a data base management system. We cannot argue with most of the suggested features and capabilities, however, we are of the opinion that currently available hardware is not yet capable of economically handling the amount of processing required to satisfy all the desired features. We are optimistic that, in the foreseeable future, advances in mass storage technology will permit the implementation of all the desired features.

As a software company, it is our objective and intent to listen to the demands of our current and prospective customers. One universal requirement is evident, no matter what data base management system a user elects, he must be able to build his data base modularly over a long period of time. He must have independence of data, hardware, operating systems, and programming languages such that he will never be required to alter previously developed programs in order to accommodate additions to or changes to the data base. A second universal requirement is that the data base management system must provide a high level of reliability and maximum ability to be recovered in the event of disaster.

Cincom's philosophy and products adhere rigidly to these universal requirements. Cincom will continue to listen to CODASYL, Guide/Share, and its own user organization for direction in its future development. The conference provided us with the pleasant assurance that we are providing our users with the proper direction.

OLSON:

On behalf of Control Data Corporation, I am very pleased to participate in this conference and am delighted at the concept. Obviously, I approach the subject of data base system requirements from a vendor's point of view. But I will not dwell on products. I think it is far more important to give you our views on user needs -- particularly those needs in view of the key issues within today's technology and what is coming in the future.

Let me begin with what we see as the objectives necessary to satisfy user data base management requirements:

1. All users must be able to selectively access a continuously available data base with their own choice of language, mode, and method -- no matter how diverse those vehicles may be.
2. You must be able to choose those data base management facilities that provide you with the capabilities, performance, recovery, integrity, privacy -- what have you -- that you need. We might call that Modular Selectivity.
3. I'll call this Evolvability. As new user-needs surface and new technology is developed, data base management capabilities must accommodate both without disrupting the user by any sudden revolution in techniques. There should be a smooth growth rather than a sudden, jarring experience.
4. The subject of standards -- Control Data is a representative on many American National Standards and CODASYL committees. And I'm quite sure we are considered as a mover behind the national and international efforts to improve standards in our industry. We will continue to help ensure that users will have an independent choice of vendors through the support of real standards.
5. We are well aware that data base management and its products create increased needs for new levels of documentation and training. We are intent on improving documentation and training toward improving the proper use of those tools that are applicable to information management systems and applications.
6. Extending data base management capabilities to meet the special needs of individual operations is of prime importance in our view. And that extension must protect original products from damage. That kind of growth must be reflected in the vendor's product where upward capability continues with product improvement or new products yet with protection of the product capabilities offered.

Now -- I must say that Control Data seems to be in line with the objectives of this conference. We all see the need for timely development and availability of quality data base management products, and delivery of those products to the marketplace.

This brings up the subject of technology. Present technology allows the User, or at least makes it possible, to select trade-offs in many areas of interest and operation -- capability versus performance, for instance. We all understand that software is costly to develop -- it is not going to get any easier or less expensive in the forseeable future either. The true requirements for security and recovery for large on-line data bases are such that present technology is strained to provide economically viable solutions to those requirements. The direction of efforts along those lines must take into consideration all the levels of sophistication of users with data base management requirements.

So what can we expect in terms of future technological efforts applied to data base management?

For one, Control Data is continuing to investigate many areas for future development -- areas that include CPU memories and features, mass memories, software disciplines and engineering principles, and the requirements of operating data bases within a network environment -- problems like distributed versus centralized data bases. And we are evaluating other considerations, including data base stations and special hardware needed to support data base management.

You might consider these investigations and evaluations as a necessary part

of product or solution evolution -- somewhat analogous to having a Volkswagen on the drawing board. Once it is delivered, we should be able to maintain pretty much the same external appearance while measurably improving the performance capabilities -- providing new internal engines and mechanics that make the product more reliable, better response time and throughput, and yet constant in terms of compatibility to the user.

Next, I would like to address myself to some of the key requirements expressed, via papers and discussions throughout this conference. In summary, I see these as the following issues:

1. Protection of your investment in developing information management systems utilizing data base management.
2. Optimization of the use of both human and system resources for solving your problems.
3. The protection of data from unauthorized or inadvertent access once placed under the responsibility of data base management software.
4. Broad data accessibility to all classes and levels of users within a system, within a network.
5. Data and system availability. The key issue is what capabilities exist for system reliability. And are there sufficient and effective recovery facilities built into and easily available to you the user to keep that data base continuously available.
6. Insurance against unpredictable information needs. The ability to grow via data independence, new applications or changed applications needs, without total reprogramming of present applications against shared data.

Following are some random thoughts on some of the major requirements brought forth during this conference:

1. Requests for more decision making tools, primarily for the data base administrator within an organization. Providing facilities for him to evaluate the user needs in structuring a data base. To be able to select and acquire system components or data base management components to best do the job of meeting the trade-offs of all users served. And tools to install, use and grow the data base management facility to serve his user base.
2. Meet the real needs of end users. There are two levels of responsibility here. One, we the vendors supply only the data management tools. You, the implementors of an information system, must apply those tools in a proper and effective manner. Then, and only then, can the needs of the real end users of information be truly served.
3. The end user facilities of the data base management system must allow for great extensibility to satisfy the changing needs of users and their requirements.
4. Requirements were specified for the ability to debug new applications or modified applications providing protection to the data base and the system. This is a critical requirement which has been poorly addressed by most vendors.

CONCLUSION

In conclusion, I would like to express our concern with some of the requirements presented here. Basically, there is a challenge to all of us. The challenge is to do a better job of summarizing and prioritizing the needs of data base management, addressing a broader spectrum of users. Isolate and identify the differences in classes of users and the variation in needs between these classes.

Some critical points in summary: We have learned of some significant and real requirements at this conference. We still need a prioritized list, bearing in mind that these items may be costly and users will have to expect to pay accordingly for many of the new data base management features requested. When the list is produced, we are still scratching the surface. Vendors are in the

business to produce products at a profit, and most users are not as knowledgeable as this group. Consequently, we must educate all users in the requirements and applied use of data base management products along with the associated value/cost issues. The only way to ensure vendor implementation is to do the following:

1. Educate users in a consistent approach to system requirements and evaluation of proposed facilities.
2. Educate users in the use of these facilities and their systems.
3. Work toward some levels of standardization within the industry.
4. Work for greater participation of a broader base of users in conferences of this nature and in systems work on data base management in the industry.

Finally, we were worried by the user experience sessions on Monday. We were assuming today's systems were not fulfilling users needs -- the papers seem to indicate they were quite satisfied. I hope we were not wrong, and later sessions have reassured us somewhat that the systems of today still have a long way to go in terms of providing quality solutions to your requirements. Control Data is committed to the objectives of providing quality data base management tools with today's technology and plans to improve these tools in an evolutionary manner as new needs and technologies are discovered.

LEFKOVITS:

I would like to say that personnaly I have enjoyed being here. And in terms of the subject I am trying to cover I would like to start out by expressing a personal opinion. First, in terms of the user requirements that I have listened to, I have not been surprised by anything I have heard. I think I have to admit, though, that I have been more than surprised by what I did not hear. I expected a great deal more to have been said of what we do not provide which I think is necessary and I think that in many ways people who have talked here have been very nice to us. I think there are a number of problems that perhaps we manage to solve today, but I think we do it in fairly unsatisfactory terms and I would have expected a much stronger statement in terms of the need to provide more fundamental solutions. There are some subjects that have been touched only very lightly and I am personally concerned with one of them: that is the subject of security. I think there have been several instances lately that very clearly point out the very great danger that the profession, acting as implementors and presumably custodians of the data, are really facing. I think the Equity Funding affair, among others, is something that we all ought to stop and think about; and we should think in terms of what does this really mean to the software as well as the hardware that we are dealing with. I think we all feel that, indeed, it would be nice if something were done about this, but unless there is a clear statement of the necessity of this, as well as the willingness to pay for it, it will not happen.

At this session you are hearing vendor responses. A vendor is in business. The reason he is in business is to make money, and if you look at the decision process the vendor goes through in terms of what shall be offered as a product, I think it is basically very simple. One says, if I have such and such a product, how many systems will I sell because I have it and how many systems will I not sell if I do not have it, and you generate the proper numbers and this basically leads to the decision as to whether it is good business to do it or not. Now unless you people come up with some clear statement that if we, the vendors, offer such and such you will buy it and will pay for it, I think there is very little rationale for vendors to offer it. It is not even enough for a few people to be so willing and I think that is an appropriate subject at this meeting because I think the people who are represented here are somewhat in the forerunner category. That is not enough. There has to be a much more universal type of demand before the process really becomes effective. I think we have to be realistic; if you ask for the greatest thing and say you are not willing to pay for it, it is unlikely that it will be provided. It is nice to be a pioneer and I am very fond of trying to do advanced things, but do not forget it was the

earliest Christians who faced the hungriest lions.

Now as to our position in the data base area. First of all, we are very acutely aware that this is a serious business. It is a deadly serious business because if you as a corporation or an institution commit yourself to the use of the data bases, you are conceivably in a precarious condition if something happens that goes wrong. I think we are aware of that, and in terms of what we offer or plan to offer we would like to be convinced that we are not giving you something that you can shoot yourself with. I think this goes so far that even if you say "Well, I do not mind taking a chance", I personally feel that still there is a responsibility: there has to be a feeling of ethics that this is not the way that we want to do things. This means that our major concern has to be the integrity of the data base. Maybe other aspects may not be quite as satisfactory, but the integrity of the data base has to be maintained. As other parts of what we feel our responsibility to be, I think that satisfactory performance must be a part of our product offerings. I think we have to hesitate to do things that are unrealistic and that the approaches we continue to look at have to be in the realm of efficiency. The last point in this area is reliability, which is obviously connected with integrity. It is reliability in the hardware area and in the software area, and we are acutely conscious of the extremely poor state of the art in software reliability. We are searching for ways to do things better and we are confident that at least a modicum of progress can be made in this area.

I would like to point out one thing that I think is appropriate to say in this meeting and that is that the tactics or the strategy that Honeywell uses might have to be different from some other manufacturers. The main reason is that we think that we have to be better. This means that it is possible that the kind of solutions we have to come up with may have to be different from others.

Now, in the area of data bases itself we have supported DBTG activities and continue to do so. It is a position that we feel very comfortable with. We feel we have accumulated a fair amount of experience with IDS, and it is an environment we think we understand to some extent. We certainly have had ample exposure to problems in the past and we hope that we now have a little better way of dealing with these problems than we did the first time around. IDS has been and is a viable system. It obviously is not the ultimate, and we recognize this, but it does represent a conceptual approach that we think is good. It is not the whole universe and we very definitely feel that the real problem we are facing is how people are going to get at the data and that different ways may be appropriate for different cases. As such, we see some programming language as being appropriate for programmers, we see other languages as being appropriate in other cases. These are end user facilities in the sense that we understand what an end user is, and also represent other languages which perhaps are used by programmers but where the decision to use this language implies a favorable trade-off - such as, if you are going to run the program once how concerned are you really with the execution time of that program. In this case, I am trying to stay away from what we call an end-user language because it may be a very procedural language which a programmer has to use but it simply cuts down on the development time of the program.

Our concern with the use of network structure files does not mean we think that what we call standard or conventional files are not important. They are very important, and I can only quote somebody's remark who was talking one day about the disappearance of punch cards with a rising sales curve. Standard files are going to be with us for a long time and I think the challenge is to find a way for people to live in this environment that includes multiple file organizations. We are also concerned with the ease of use of data and with the whole problem of small user versus big user. I will not elaborate on this issue, but some people have more resources than others and some peoples' problems are different from others. Please let us not forget that there are many people from small shops who simply do not have the time to set up committees to reach decisions on whether one particular approach is better than another, and I think that we as a manufacturer also have responsibility toward people like that.

Obviously there is a question of trade-offs and the more input we get from user groups then the more information we have as to where it is sensible to offer trade-offs. Hardware is obviously part of the whole picture, and as the situation here gets to be better and better developed I think we will see more hardware playing part in the data base problems.

The kinds of solutions and technologies that we are looking for have to be long lived. I am not sure which previous speaker's figures I am quoting that it takes three years to develop a system and the expected use of it is seven years. I think we cannot afford to come in with experimental or untried solutions and after some time find out that they did not really work out as well as we thought they would. This is doing a great disservice to the user and we have to be conscious of these dangers and react to them. Thank you.

EDITOR'S NOTE:

During the opening session of the conference, the conference chairman presented a light-hearted overview of the program which included visuals intended to evoke appropriate images of various aspects of the program. Amongst them was a picture of a dozen overfed and lumpy walruses lounging on a rock. This image was associated with the assembled collection of vendors. Thereafter at the conference and perhaps forevermore in the data base world, "vendor" came to be identified with "walrus".

Subsequently during discussion Dr. Lefkovits found occasion to make note of the parable that "the earliest Christians got the hungriest lions". These disconnected incidents were neatly coupled by the following speaker's initial remark when he stepped to the podium and gazed out upon the large pride of users in the audience.

REINSTEIN:

'Tis the plight of a Christian Walrus.

. . . . .

A great deal of what is being said this week has come under the heading of "long-range requirements" and what we felt we would do is to comment on those requirements as we understand them.

The requirements have, in fact, mostly been stated at this conference, and from places like the SHARE Data Base Project.

To give some structure to this presentation, we have divided the requirements into four areas. Installation and Management Control is the first area.

Installation and Management Control, as we understand the requirements you are giving us this week, emphasize usability aids which include models, generation aids for data base statistics, and generators of various kinds in the areas of planning, designing, monitoring, controlling, and modifying data bases.

Data Administration, one of the two things which we have felt for some time are the key areas in DB/DC Systems, had in this conference the typical characteristic of defining structures and formats, as well as the declaration of rules concerning data; for example, checking, range checking, and so on. But, in addition, you have stressed a third concept, which, I must admit, we had not considered as much a requirement as has come out this week. I am referring to Control of Storage Mapping. Your comments concerning controlled redundancy of data as it is stored have been most interesting, and whole we consider that it is a valid requirement, I want to point out that it is a solution to some other problem. That problem, in fact, is not that the data is stored non redundantly; it is some other problem like performance, integrity, or geographic distribution. As such, I think that while we need to provide to the user of a data base management system the ability to control redundant storage of data, it should at no time be apparent anywhere

but at a data administrative interface. In particular, it should not be apparent in the application program.

In-situ testing was a concept brought out at this conference and this applies, I think, to the installation as well as to the application programmer. This hasn't been as strongly stressed, but the concept of having a data base administrator with his control and responsibility implies that he too be able to test what it is that he is doing and the kinds of effects he will have on the system, just as application programmers need that capability. What we need now is a better understanding of the Data Base Administrator's "test mode" requirements.

The next category I want to address is Application Programming. In the area of application programming, something that has been coming across very loud and clear (and, if you can remember back to what I said earlier, that there were two key major items that we felt you were talking about as requirements of DB/DC Systems, one being Data Administration) is data independence. I'd like to state simply what we understand such a requirement to mean. I think the operative term is "change" and that data independence is the means by which you believe you are going to achieve a lowered cost to your installations in the presence of change. There are at least two kinds of change that you talked about this week, and I think they bear mentioning as specific items or kinds of changes. They both relate to data structures. One is changed to the organization (usually stored) of the data. We understand an extremely real requirement that when the organization of the data changes for whatever reason, geographic distribution, restructuring for performance, whatever, when the organization changes you do not wish in any way to have that change reflected back to the Application Program base, or incur cost in that area. We accept that requirement.

A second area of change has to do with the Application Program need for data. We notice that in an installation it is very often the case that the addition of new Application Programs eventually gets you to the position of having the N+first program have data requirements different from, but overlapping with, the previous N. That particular kind of change in the composite needs of the application base should not cause, in any way, a reprogramming cost or any other cost to the original or existing applications. These two things have been discussed rather at length at this particular conference, and we recognize them as requirements. There is, however, a third requirement for data independence which has not been addressed at this conference, and should have been. It has to do with the subject of "intent" as it is meant by the Application Program in dealing with data. It is our understanding that when you talk about data independence, you're talking about a requirement that there be the capability for extensive data sharing between programs. We have observed that this requires, not only the ability to map structures from the application level to the stored level, but also the ability for the system to understand the intent of a program while modifying the data base. An example of this could be that although it is quite possible to write an Application Program that goes through a data base, performing deletes by putting "5 z's" in the front of field no. 2 (such that when it goes back through the data base at a later time detecting the "5 z's", it knows to bypass the record), this kind of programming masks the intent of that particular get and put to of the data base system. As such, it precludes effective sharing between different Application programs and affects data independence.

I would like to state our position on multiple views. We do, in fact, recognize a real and valid requirement for a data base management system to support multiple views of data. Specifically, there are three that should be supported in a data base system. They are IMS hierarchies, networks, and relational views of over flat files.

Data dictionaries are obviously a requirement and in fact relate to data independence.

You talked about integration of data communication with the data base system. With regard to the relationship of the data management system, and the operating system, you have mentioned that most DB data base management systems will operate in an environment of parallelism and multiple users. In most architectures, the

concepts of resource management and parallel execution fall to the operating system, not the data base management system. Your requirements dictate that the marriage chosen between the data base management system and that part of the operating system must be one such that there are uniform and consistent interfaces for work and resource management across the system.

I think we are hearing a requirement for a shift from procedural orientation to declarative orientation. This isn't theoretically necessary, but in a practical sense, we find this is the best way we know of to try to insure that the system itself can know intent, and can have enough information to guarantee consistency, integrity, and reliability of the data. So, I think you will hear us responding to all of your requirements with this kind of a shift.

You have not talked, and this is something I would like to criticize you for, about the relationship of the data base management system to the languages. (There was one comment, that some of the COBOL functions should be in PL/I and that's a fair comment.) There has not been, however, the kind of discussion that I would have expected as to what you think the relationship of data base management systems are to the current problem oriented languages, COBOL, FORTRAN, and so on. Those languages will exist in the time frame that Dr. Sibly has addressed (and that we are generally addressing with long-range requirements), and I don't think that today's relationship of those languages to data management as exists in most systems is appropriate. We need to pay special attention to the issues of "intent", and array structures as they are supported in HLL's vs. how they must be supported in DBMS'.

Data dictionaries, queries, prompting, future languages, and terminal support are a few of the things you have addressed at this conference and relate to the end use area of requirements. Again, partially in answer to Ed Sibley's question, I think that systems of the far future (however far that may be), are going to see their biggest growth capability, and the highest concentration and orientation of their features in this area, and not in the others. In particular, user languages are going to become an extremely critical part of the data base management system. I do not believe that it is going to be appropriate for the vendor to supply all or even necessarily a significant number of the user languages. I do believe that you are saying it is a requirement for a system to be properly architected so that user languages can in fact be defined by other than the vendor and by the installation or whoever is appropriate in a reasonably easy and convenient fashion.

Terminal support (data communications) is the way into data base and they go hand-in-hand. In particular, I want to mention something about graphics. There has been a lot of talk in data base management systems as to the peculiar requirements of the graphic data base. There is, in my opinion, far too little in the way of formal requirements in this area. I personally do not really understand the unique needs of the graphic data base as different from the file structured, record structured data that we now talk about. I intuitively feel that there are differences because people I respect very much tell me that, but I do not understand it in the sense of what it means to system architecture.

In general, recovery, security, education, and the others that are listed on this particular page are issues you've brought up in varying degrees at this conference. One of the new concepts in recovery, I think worth mentioning as a requirement coming out of the user community, that is different from past systems, is the concept that errors are going to be detected by Application Programs at times quite later than the actual error occurred. This is different from the philosophy with which operating systems and data management systems have been designed in the past where we have assumed that the system detected the error at some time and that we could at least arbitrarily call the occurrence of that error. With regard to the relation of batch and on line; this too is something different from past philosophies. You've said fairly clearly that you want schedulability of batch operations from the on line operations and you've said very, very loudly that the presence of an on line DB/DC System does not preclude the necessity for a good batch system.

Migration you haven't talked about, and I wish you'd have talked about it more

because I think that there is a tremendous number of requirements in the migration area that we are not hearing.

Broad spectrum of support is another "requirement area". I mean by this two things, and again this is an area you have not addressed very much, and I wish you had; first of all, it is very easy to respond to long-range requirements by trying to say "Yes, I've heard everything you've said, and we are going to be all things to all people and that's the best way to satisfy long-range things." I've tried not to say that. I've tried to in some ways constrain, or put some bounds on some of the things I've listed on the previous pages. Most of those bounds will probably come right here because it is in fact going to be the case that any data base management system that meets the entire needs and requirements that we are talking about this week is not going to be delivered to the users in one single monolithic package. You're going to get it in pieces and stages because we are talking about extremely extensive systems, and if for no other reason, you're going to get them in pieces because we've talked about how it takes you from 2 to 8 years to implement your system and I doubt you want the entire set of functions there at first. We do not understand what makes a coherent strategy or a coherent plan for the order of functions as they would be presented in any single installation. You, yourselves have said this week that you intend your own development of data base systems to be staged over the many years you put your resources into it. You must also, I think, think more about how you intend to do that or what you would like to see from the vendor in the way of functions. Additionally, there are "small users" and where as before they were measured in dollars, I suspect now they are going to be measured in things like transactions per second or by size of data. We do not yet understand their requirements.

SCOTT:

INTRODUCTION

I shall try to describe NCR's philosophy and intentions with regard to data base in the context of our overall business strategy, both from the standpoint of our on-going strategy, as well as from what requirements we think we have derived from the Conference.

Then I shall comment on some of the specific requirements put forth in the papers and discussions. Finally, there will be come suggestions to the community regarding our future priorities.

NCR DATA BASE PHILOSOPHY

NCR's DBMS philosophy is meaningful only in the context of her overall systems philosophy which, in turn, is derived from her marketing strategy, NCR priorities are (1) efficiency (we mean small and fast -- in that order) (2) reliable and, especially, (3) easy-to-learn hardware and software systems. In all cases, we must stand ready to provide continued growth for customers with expanding needs. Since the smaller businessman can afford to pay less for added capability, software improvements must be highly compatible from one release to the next in an effort to minimize conversion costs.

This philosophy extended to data base management suggests that we provide products with capability similar to TOTAL as opposed to IMS, for example. Our company requirement for simple, easy-to-learn software is particularly difficult to meet in a DBMS since the state-of-the-art in data base is complex, ambiguous, theoretical, and esoteric. A lot of our current DBMS design effort involves simplifying or clarifying technical or aesthetic deficiencies in the systems offered or proposed today.

Our objectives, in priority order, for ongoing development of data base systems are:

1. Reliability/recoverability

2. Programs independent from stored data
3. System simplicity (easy to learn and use)
4. Performance (space efficiency, time efficiency)
5. User conversion problems
6. Capability

We feel that the medium size user has the same needs for improved data access, security, centralization relatability, and so on, as the large user. We currently have special interest in the DBTG specifications (as evolved by the DDLC and DBLTG), and we would like to develop an experimental system which hopefully will validate empirically that the DBTG architecture satisfies the needs of our marketplace in terms of performance, capability, and ongoing cost effectiveness. In addition, we hope to continue our design to a more advanced data management system envisioning some of the requirements put forth here.

RESPONSE TO USER REQUIREMENTS

As for responding to the general requirements stated here, we have distilled a pattern of five very important needs:

1. The first priority requirement seems to be the independence of programs from storage structure and access method, physical format, and the viewpoint of the DBA and other programs. In response, we too feel this is high priority (2 on our list) but we are somewhat concerned over its efficiency ramifications and the problem our customer will face converting to it.
2. The second requirement put forth appears to be system reliability and the capability to recover -- this is our #1 priority, and we are concerned that the current technology amounts to a bag of tricks and tools dumped on the DBA.
3. The third center of attention appeared to be time efficiencies, especially for on-line use. This is a high NCR priority but given the classical time/space trade-off, we are in a bind because most of our users run on small machines. Consequently, our DBMS is designed to function correctly on small-memory configurations, but to run faster with each increment of available memory.
4. Ease of use was a predominant theme, both for programmers and non-programmers. Our response to this requirement is twofold; first, we feel that bad documentation contributes more to difficulty than system complexities. Therefore, NCR has recently embarked on a policy of spending time and money both on internal and external documentation. Our second statement is that we have no immediate plans for a facility of the power described by Pat Nichols since we have neither the technology nor resources to realize it. We would like to move in this direction.
5. The fifth requirement we see is flexibility, particularly with regard to advanced data structuring and access features. Not only is this important but it's inevitable. However, we shall move slowly and carefully in this direction because the industry has not yet converged entirely on the required features, nor have they agreed on an acceptable interface to those features they have agreed upon, nor has there been any empirical proof that these features can be implemented efficiently.

In conclusion, if we were to make some general suggestions to the data base community, I think that we would ask that our priorities be rearranged slightly in two cases. First, we would like to see more emphasis on immediate-term rather than long-term objectives. In other words, what should the vendor do in the next six months to two years? A couple of papers separated their short-term needs from their long-term, but the requirements must be made further definitive (e.g., "reordering" group types, or "splitting and combining" records as opposed to data independence") and then prioritized so we may implement the most important first.

Our second suggestion is this: we would like to see academic priorities shifted to some of the more ambiguous, but perhaps less intriguing, technical problems of data management. For example, a study of data base and program recoverability in a concurrent update environment or data interchange, or file conversion, or restructuring techniques will benefit us much more than further insights into deadlock and/or randomizing algorithms or overflow techniques.

In other words, we are suggesting that our research be calibrated against the potential economic payoffs to users which we feel appears to be in keeping with the underlying philosophy of the user requirements presented in the papers here.

HARTLEY:

Let me begin by first expressing my gratitude for the opportunity to participate in this conference. It has been a most productive and interesting four days.

What I should like to do in these few minutes is; first, quickly review what will be an obviously incomplete list of the "needs" which I hear expressed at this conference; next, make some comments as to how I feel these needs will be met by Sperry UNIVAC and by other Vendors; finally, put in a play for increased participation and support of the CODASYL Data Management standardization activities.

Turning first to the user needs in the Data Management area which have been discussed at this conference, I have reviewed my notes and made a list of those which I could recollect. This list therefore unordered and incomplete.

Thy needs which I have noted are:

- need for increased system stability
- need for increased data security
- need for distributed processing capability
- need for input validation by the Data Base Management System
- need for data dictionary
- need for standardization, to increase portability and reduce conversion costs
- need for a greater degree of data independence
- need for end user (query language) facilities
- need for better statistical monitoring tools, tools which not only tell the Data Administrator what is going on, but what to do about it
- need for rollback and recovery capabilities
- need for reorganization and restructuring capabilities
- need for telecommunications facilities

It is my opinion that, to a large degree, everything on this list is within the present state of the art. However, I don't know of any existing Data Base Management System that meets all of the needs on this list. I also don't know of any User who is prepared to use all these facilities if they were available, or if he possibly could, would have sufficient processing time remaining after their execution to actually get the days work done. Furthermore, just because these facilities are within the state of the art does not mean that one can expect to see them in the next release of DMS 1100 or any other Data Base Management System.

The point I am making is that Vendor development of data base management software is an evolutionary, not revolutionary, process. I think this is obvious for much the same reasons that the introduction of a data base and the development of applications around it in your own shops is a lengthy, very involving and evolving process. A second point, less obvious, is that a Vendor cannot pick one of the above mentioned needs and develop it through to its completion (a somewhat arbitrary point anyway), but rather a Vendor must move forward on several fronts at once, thus delaying a total resolution of any one need.

In my view we can expect new hardware to help speed up this evolutionary development--not because it will be faster and cheaper, although that will be important--but because the hardware will become more capable of performing data

base management functions. These functions will very likely appear in the mainframe, in data base oriented mass storage devices, in specialized data base management machines serving as an adjunct to the mainframe and in intelligent terminals.

It must also be kept in mind that Data Base Management Systems, as they evolve, will place more and more responsibility on those of you who serve as Data Administrators. This is because many of the above needs will be met with facilities having a series of options. For example, in yesterday's discussion of Data Independence I didn't hear any conclusions--and I didn't expect to--as to what Data Independence is, when it should be done, to what extent, at what cost, etc. As a result Data Independence is a good example of a facility that will likely be offered by the Vendor as a series of options providing for different degrees of it at different times (compilation, collection, execution, etc.). You might respond: Great! At least we have it available, we can make the decision as to how to use it. However, it will now mean that there are more variables which must be analyzed, more possible permutation of the system, more trade-offs to be made, more decisions for which you are accountable. As a result your job may not be made that much easier.

One final reaction I have regarding what I have seen or heard at this conference is that some of the requirements expressed by Users, particularly in the discussions, run counter to those which I hear when I approach Users in their shops. For example, the Vendors have been critized--perhaps rightfully so, I do not wish to argue that point--for suboptimization of Compilers, Sorts, Data Base Management Systems, etc. and not optimization of the overall system to meet the User's data processing requirements. However, the Requests for Price which the Vendor receives, the benchmarks he is asked to run, are very often suboptimal measurements of transfer rates, instruction cycles, compiler speeds, etc. which are not accurate reflections of actual processing requirements. In short, Vendors are often asked to perform one way and measured another. Along the same vein, Vendors have been critized in the discussions for locking the Users into their systems--again perhaps rightfully so, that is not the point--the point is back at the shop the Vendor is often under pressure from the User for features which increase efficiency, but do so at the expense of hardware or software independence.

In closing let me state on behalf of Sperry UNIVAC that we feel that User/Vendor interaction, such as has occurred here this week, is absolutely essential. No-one, ourselves included, has all the answers, or is even aware of all the problems associated with Data Base Management. That is why such interaction is so important. This is also why we fully support the work of, and actively participate in, the various CODASYL Data Base Standards Committees. These committees provide a forum for interaction on a continuing basis and we at Sperry UNIVAC urge you to participate with us in this very important work. Sperry UNIVAC's Data Base Management System, DMS 1100, is itself based upon the CODASYL Data Base Task Group report.

PIERCE:

During the past three days we have heard some conference participants describe their requirements for a DBMS. The statements have ranged from expressions of general requirements, e.g., "better system performance"; to very specific requirements, e.g., "Move fields between segments...in an IMS data base." Unfortunately, most of the discussions have come from representatives of large installations. I do not mean to imply that we should therefore discount the stated requirements, but rather that we must add to the "list" the requirments of other types of users of a DBMS. We have not heard from the user that has had little previous experience with a DBMS, whose data base is not 2 billion characters in size; but nevertheless has a definite need for, and thus requirements of, a data base management system. I believe this was pointed out by the statement of one participant who in effect said that she was a small user and her needs were different and really seemed trivial in comparison to those stated.

I believe the requirements stated have centered around three main issues:

1. Data/Program Independence
2. Data base Administrator Aids
3. Data base Integrity or System Reliability

The previous discussions on data independence have been somewhat cloudy as we have many different definitions of the term "data independence". Most of the discussion, and at times argument, has centered around whether the DBTG type specification of data structures is sufficient or whether there must be a higher level of specification, the "Relational description" of data.

I will agree that an installation with a 2 billion character data base and 3000 production programs may not have enough time in the day to restructure the data base and recompile. That such an installation could better afford the additional execution time mapping that would be involved in a third description of the data structure. However, there are many installations that would be further ahead with an occasional restructuring of a smaller data base. It would appear that if we are to satisfy the requirements stated for data independence, then the current CODASYL specifications must be extended. We feel this can best be accomplished by our working together on the various CODASYL committees to bring about the needed extensions.

We have made an attempt here to define the role of the data base administrator and to state some of the tools he needs to fulfill his tasks. This conference seemed to indicate that his primary requirement was for a "data dictionary". I do not find this to be the first need of the installations I am in contact with. It is my experience that the DBA must have assistance in structuring the data base and utilizing the DBMS to efficiently process the companies' data. I believe this difference in requirements is primarily caused by the size of the installation which directly affects their prior experience with data management systems and the funds available for current training.

Much has been said about data base integrity or system reliability. We have heard requirements for systems to test new application programs for an existing data base; for performance monitors and tools to determines system or data base availability; for procedures to allow temporary modifications to data in order to assess the impact of the change. These and most of the other proposals for data base integrity are certainly feasible. My only question would be whether most users could afford the cost of such a system. Thank you.

EDITOR'S NOTE:

Ochel (MRI Systems Corporation) chose not to submit a written response.

Data Base Management Systems, D. A. Jardine (ed.).

PANEL DISCUSSION NO. 1.
TECHNICAL ASPECTS OF DBMS

Panel: Charles W. Bachman
Robert Engles
Peter Hill
Donald Jardine
A. (Tax) Metaxides
Kent Ochel
Jon Turner, Moderator

TURNER:

The first topic for discussion is: do we have the capability for both logically and physically distributing an integrated data base? What capability exists or is required to distribute on the basis of usage statistics?

OCHEL:

I believe that we have the technology available to do that. There's no question that we can physically separate data; the major problem is the logical problem, when you subset or stratify data. The association of the lower level of data to the higher level of data is logically not defined yet nor is the control process to handle it. The concept is that of summary data at the higher levels which reflect the lower levels of the provincial data base. The methodology for doing that in an economical manner as well as a methodology for controlling those relationships has not been developed yet.

BACHMAN:

In the case of a manufacturing control system that represented Honeywell factories in Phoenix, in Boston, in Europe, it would be quite reasonable to think that the basic manufacturing information ought to be at the site at which it is being used most frequently. It's quite reasonable and within our technology to integrate those data bases in such a way that a program could search through certain logical structures, could move from what's in Boston, reach out for what's in Phoenix or what's in Paris, extract that information, and continue searching through logical structures which in fact were physically distributed. There are only some questions of performance and economics that hold us up. When the business requirements need this, and that requirement was only faintly echoed yesterday and the day before, the systems will be delivered to do it. It's not a difficult technical problem.

METAXIDES:

I think that more research is required before we can solve the economic aspects of the search that Charlie Bachman was talking about just now. I think there are some situations where it would be relatively easy to logically and physically distribute. That depends on the particular data base and on the data base design involved. In other situations where they're highly integrated, I think we'd have some real optimization problems that, at least to my knowledge, have not yet been solved. It's in part a question of modularity. I think we would want to minimize the interfaces between each of the distributed portions of the data base.

JARDINE:

One of the interesting problems that has been alluded to a couple of cases already is that of consequential update in another data base and the control of integrity of the entire distributed data base under those conditions. I'm

thinking of the case where data base A has a field that is updated and this triggers the update of other fields in other sub-databases in order to maintain consistency across the distributed data bases. If we have a distributed data base, who has or where is the responsibility for control of the overall consistency and integrity of the data base? If this is distributed among the data bases, then I find the multiple update problem hard to understand, and I think we have to break that apart very carefully. If there is a central control, then I'm not sure we have the kind of distributed data base that people are talking about.

HILL:

I think that I agree with Charlie Bachman in that it's well within the technological capabilities to provide this facility today. I think one of the toughest problems to crack, an extension of what Don Jardine said, is not just the control of the data but the ability to handle the problem of data integrity and recovery in an environment like this.

BACHMAN:

I think you can separate this problem into two parts. One essentially is data storage structure and we've got very clear indication from this audience they want program independence from storage structure. Now if the storage structure happens to stretch out across the U.S. or across the world, it's not substantially different from stretching it out across several different spindles. You have a reliability problem, true, you have a communication problem, but you still have to get to two different spindles. That's a communication problem that we've solved very well.

If there is an update of one field which impinges by declaration on some other field or set of fields, then I think it's appropriate the data base management system should respond and handle that problem whether it's on the same disc pack or not. I think that's a separate issue.

OCHEL:

I think that the update problem may not be as bad as we're saying here because very soon the technology is going to allow for the concept of the virtual field, very economically. Therefore one update of a given field may well suffice for all the remaining interrelated fields in that the interrelated fields can be handled virtually rather than physically.

TURNER:

There have been some interesting experiments by various network projects investigating what changes to data management systems were necessary to keep track of where data was physically located, and attempting to determine the best routing for the data. The ARPA network I believe has some project reports on those experiments.

BASH:

The response time of a distributed data base may be quite different than that of a local data base. Switching from one spindle to another can be done in a matter of milliseconds. Switching from New York to Tokyo takes much longer. As a result of this, we probably will take a distinctly different tack towards optimization. Because of the cost of disk storage we have been making a great effort to reduce redundant data. When I start talking about data bases in New York and Tokyo and London, data which is heavily used in all three locations will probably be redundantly stored. The concurrent update problem may very well be the major technological problem. We may have to be able to keep them simultaneously identical. They have to be re-updated at some later point in

time. Thus the data base does not have full integrity at any given point in time.

BACHMAN:

The option to physically replicate the file at several different locations is an important one to remember, and this will be one of the decisions open to the data base administrator. He may choose to maintain it in a single place and keep it up-to-date for everybody, or he may distribute copies of it which will be updated daily in some other way.

ENGLES:

Given a distributed data base, the interface between an application program and a DBMS should be reconsidered. Suppose you have an interface of the type where the application program is saying 'Get Next', which gets at most one record for each request, and the program is trying to select n records. This can be inefficient if each single request has to be broadcast across the network. We might then reconsider this kind of interface and give the application program the ability to request the n records that it wants with one request.

MAIRET:

I'd like to question Charlie Bachman on the statement that switching from spindle to spindle regardless of their locations is just a communications problem. I believe there's more to the problem. There is probably also a data base manager and another machine in that far-removed site who is accessing and maintaining that same data. If you're talking about sharing data on a distributed network and there are multiple data base managers, I think there is an additional problem: who is in control of what? Is there a new concept of master DBM's and slave DBM's and if so which one is master and when? How do you control that?

BACHMAN:

I think we must consider a data base system with distributed software pieces supporting it. There has to be a formal unified control, and there are many possible strategies for doing this. If you're going to lock out a record, it's got to be locked out to everybody, regardless of where it came from and who is asking. There has to be unified control to make it work.

METAXIDES:

I have done some work on the question of distributed data bases and also on the problem of allowing distributed applications and distributed data base management systems. A possible arrangement is the following: applications reside on various central processors which may themselves be large scale or mini machines. All application requests for DBMS services go through a mini which handles all data conversions and serves as a load balancer in distributing those requests to one or more data base machines. The data base machines in turn interface one or more data bases via a mini which performs data coordination functions; that is, it handles concurrency and locking problems and distributes requests to the appropriate data base. Architecturally this approach seems reasonable. However, the economic aspects, the transmission and optimization problems are subjects which will need a great deal of work.

OCHEL:

If you try to concentrate the control through a single mini the availability goes down when that mini goes down.

METAXIDES:

If the mini is very small it's quite possible to have one or more backups for it.

MOEHRKE:

Responsibilities for file maintenance activities are usually concentrated in one functional area. It's not unreasonable to expect an evolution by way of redundant data bases which are software coordinated. This is our approach so far. File maintenance may occur at an appropriate time, maybe once a day, maybe once an hour. This is not what we need in the long-range, but it certainly is a feasible short-range way of doing it.

HILL:

I think you can go further than what you suggest. I really don't think that data redundancy is required to perhaps the degree you indicated, if you use redundancy of structure. In the sense that you use the same structural definition throughout the distributed data base, where in fact the occurrences of the data may be in need at different locations at such a network. It's a very reasonable thing to accomplish today.

METAXIDES:

The more information you can give the DBMS at one time, the greater opportunity you give it to optimize. The whole idea of starting off with selection expressions that relate to one record and of having selection expressions that work with a general search command is that the list of selection expressions can be open-ended. This permits complex selection criteria as selection expressions and a named set as a result. The resultant set can then itself be operated on by any of the selection expressions. I think that is the kind of interface which Bob Engles is suggesting is required, and I agree.

BACHMAN:

A 'Next' command is not really any different than a 'Read' command. When we see a 'Read' command, we expect to get the next 'Read' command behind it. It's certainly reasonable that data base management systems should anticipate the following 'Next' if someone says, 'Retrieve the next record of a set.'

GOSDEN:

When you consider the control of a distributed system, the good thing to do would be to look at the way we control distributed non-EDP systems. Very large scale systems that work well are the ones that seem to map the management types of controls, responsibilities, and hierarchies that we have in the real world today. There's a huge variety of those and what we need are different decision structures that we can build comparable command structures inside the data hierarchy to the management hierarchy of the organization.

BACHMAN:

I think you were describing a <u>decentralized</u> organization, and contrasted <u>distributed</u> organization.

GOSDEN:

We need both, and we need all the mechanisms of both, because every organization is partly each in fact.

TURNER:

Let's pass on to another group of questions.

1. Is the world flat, a network, a hierarchy? Should the world be modeled as flat, a network, a hierarchy?
2. There are two basic approaches to the way that data base management systems handle structures. Networks, such as those proposed within the DBTG systems, and linking trees such as IMS. What are the pros and cons to the user and perhaps to the implementor?

BACHMAN:

We should start by saying that the ideal solution is to model the world as it is. The world itself is flat or hierarchical or network, as a particular person seems to view it. People's views are evolving, in this sense. My thesis of the other morning is that the world is structured as a network, and that we all have many relationships with other people, with business, with cars, with property, with concepts associated with us, and that these concepts, cars, people, and houses are all associated with other people. In fact, we have a very complex relationships which you can call network if you like.

JARDINE:

The problem of modeling the data structure after the real world is that of deciding how realistic a model to incorporate into the data structure, versus how realistic a model to incorporate into the application program, versus how realistic a model to incorporate into the user's view of the results. It was pointed out to me yesterday that the derivation of the fundamental model of the real world, which I agree is a network, and where we place it in the whole range of activities going all thy way from the user's interpretation of his results to the stored data structure, is an extremely complex, difficult, and possibly the fundamental problem of data base architecture.

HILL:

I'd like to comment on the second question. IMS offers physical tree structures both in a storage and logical sense. It offers interrelated tree structures in a physical sense, but always still views these in a logical sense as a tree structure. It is our opinion as pragmatic implementors, that this technique offers the flexibility required to do many of the things networks could, but to do them in a highly efficient fashion, and in a manner that was easier for users at the application interface to understand then if they were viewed as networks. It happened to be the preferred choice and one we felt represented a large portion of the world's needs.

ENGLES:

I've a theory that during the Middle Ages the world was, in fact, flat and that it became round since then. Today, hierarchies, networks, and flat files, are equivalent. They all carry information equally well. Let me use the term relational model, instead of flat files.

From an information point of view, the network model and the relational model are equivalent. With the relational model, you have fewer names; the only thing you really have are record types. These record types are what you name. Ted Codd calls them relations. With the network view you name these record types and you name the connections between them, the DBTG set. From a pure information point of view naming sets is an unnecessary thing to do. I think it can be useful to name relationships between record types. What do we wind up with? I think that a DML for a relational model will be simpler and slower than one for a network model with today's technology.

METAXIDES:

I do not equate networks with complexity. We are all able to see in our every day lives that one thing is often related to many others, both in a parental sense and in a dependent sense. For example that is quite evident in an IMS data base. Whether you call them interrelated trees or not, they can also be called networks. The question now turns to what the user of such structures should see. Given a network, a user that wants to look at particular trees out of that network can do so quite successfully. On the other hand there are some situations such as bill of materials, parts explosion, and PERT charts, which are true networks. There is no way satisfactorily to deal with them, to be able to stand at one node and look forwards and backwards simultaneously to n levels, if you regard them as trees. That is a serious problem in a situation where the user is given only tree capability. Turning to the relational approach, I believe that there is very little difference in terms of the end user between the relational approach and an approach which provides structure. In the relational approach, we talk about normalized relationships, such that there is no structure within a relationship. All of the structure has been flattened out, all of the commonality has been put back in; there is no factoring out. For example consider a simple tree of state, county, city, building. We would have a record at the bottom levels and each of those records or relational n-tuples would have state, etc. in them. It would appear on the surface that such an approach would involve much repetition of data. However, one does not implement it this way. The implementation could actually be very similar to that of the network approach. From the user's point of view there is no reason why one cannot take a relational view of a network structured data base.

ENGLES:

I agree with your last statement but only if the DDL is capable of saying what is the basis of a relationship, in other words what are the fields that form the basis of the set relationships, such as department code in the employee record.

BACHMAN:

I think it's quite clear to anyone who has looked at the DDL's that support this kind of structure that you very carefully declare the field which controls the set relationship. Further, you declare the fields which control the ordering within those set relationships.

ENGLES:

What then is set occurrence selection for?

BACHMAN:

Some of the options allow you to control this manually and while manual escapes exist, they are definitely declared for totally automatic action. I'd like to go back to a question that I wrote down myself after having a very long and interesting discussion with Chris Date at breakfast yesterday. We were trying to describe this very issue that has been stated, the relational view versus the set view, and the question I wrote myself was this one. I think it's a dilemma and I don't think there is an answer to it. Are the sets derived from the commonality of attributes (the Codd view), or is the commonality of attributes derived from the natural sets that exist in the real world? I don't think there's an answer to this. I think both co-exist and you could choose either view you want a prime and build a system on it. IDS, DBTG, and those which are based on sets, think the set is the natural thing and the attributes are a way that you hold them together in a typical paper system. You can choose the other view and

I can't say you're wrong.

OCHEL:

Metaxides indicated that he didn't really think a network was any more complicated than a linked hierarchy from a practical, pragmatic point of view. If you consider a user language for a network versus a user language for the much more disciplined hierarchy, I would have to disagree. I think the network is definitely more complex from that point of view. The disciplines imposed by linked hierarchy permit a much easier development of user language capabilities. From a pragmatic point of view, it is more reasonable to consider a structure that is a linked hieratchy, rather than a network. Until we develop the appropriate disciplines for networks that are acceptable to the world, we should approach it in this manner.

JARDINE:

This is a clear case of putting the understanding of the data model that we have built in the hands of the user, rather than in the application program or in the data management system. That may very well be appropriate. The user understands that the model is a hierarchy and he interprets the data in that sense. Hierarchies are easy to understand. If you put the understanding and analysis of the data model of the real world into the hands of the user to interpret, and you then make it a complicated network, then he cannot understand what is happening.

SEVCIK:

With reference to Charlie Bachman's statement about modeling, the biggest advantage of models is the freedom of choice. The distinction between the model and the real thing is the freedom to choose elements that should be represented and leave out those elements that need not be represented.

Secondly, Tax Metaxides' comment that the relational view necessitates a lot of redundancy is false. There is freedom of choosing the relationships as you wish, and if the relations are chosen appropriately, the redundancy that you implied does not exist.

METAXIDES:

The redundancy is only logical. If I have to express the semantics of a relationship between two relationships by repeating a common identifier field, that is a repetition and redundancy. I believe that is a necessity of the relational approach. The only difference I see between the relational view of data and a network-oriented view is simply that one is much richer than the other in semantic content. If I was given a complex data base expressed in relational form, the first thing I would do, because I see all of these separate relationships, is to try and see how they relate to one another, by mapping on a piece of paper for myself the overlap between them. A network already has that semantic aspect built into it. That is the only difference I see between the two. I think it is an important one.

BACHMAN:

In answer to Ken Sevcik's first comment, the question we considered is what should the modeling tool allow? If a modeling tool allows networks, it turns out hierarchies are subsets of networks. If you want a further subset you're back to a flat file. The critical question is what will the modeling tool permit the modeler.

STEEL:

The model of the real world that we use, whether a hierarchy or a network, is a mathematical concept. How we choose to physically represent this in some kind of manipulation mechanism is something different. We ought to very carefully make that distinction. If I want view the world as a tree, I ought to be able to do that. If I want to view it as a network, I ought to be able to do that. From a mathematical point of view, I can think of the world as consisting of things and sets of things. I can think of relationships between things (and things can include sets of things) as being sets of ordered pairs of things. I can construct out of the simple concept of set membership, the notion of ordered pair and I can construct out of that sets of ordered pairs. I can define properties of sets which allow me to speak, for example, about orders in the sequence. All that is mathematics. It has a certain amount of respectability and a certain amount of history. None of that speaks to how I should actually put things together inside a piece of electronic machinery. I think it's very important that we keep these two ideas separate. Implementation is one thing, but a model is something quite different.

BACHMAN:

We have proposed a series of mappings from the end user's view back to a recording on some media. The real capability we're looking for is the ability to move up to the virtual structure, where we're free to view the same information in multiple but consistent ways for the benefit of a particular person who wants to manipulate the data. That has to be reflected back through various mappings, back to the logical representation, back to the main core representation, and eventually back onto the media.

METAXIDES:

In a storage structure we are dealing with strings of characters or bits. All data structures, however we look at them logically, have to be reduced to that. From the users point of view, I would like to be able to work with a hierarchy, a network or a relational data base. All of these are quite compatible with one another. Given the concept of sets with names, suppose I want to reduce that, with the present DDL specifications, to a relational view. This can be done by means of the source declaration, which doesn't actually create anything. Rather, it takes the storage arrangement (which we have already said can be the same in all cases) and collapses the hierarchy, pulls it all back down to one level, in effect creating a series of n-tuples The programmer can then use that virtual record. The DBMS will be doing a lot of work but that is the case whether you only allow a relational view or whether you allow the other views. As a user I have to say that different people see things differently. From that point of view, we should tend towards and build systems that will accommodate all these views. There is no basic dichotomy, no basic disagreement, among these views.

LOWENTHAL:

Several speakers have said that the user needs to have choices as to how he reflects the real world in his data structure or storage structure. I find that the user is not particularly interested in reflecting the real world in his data structure. In fact he is willing to sacrifice that correspondence in order to minimize costs. He will design his data base to effect certain trade-offs that optimize access times or storage costs. Intellectually he translates the results back to the real world. Maybe in the future he will either decide that it is unnecessary or else the system will do the transformation for him. However, in the short term and I suspect for many years, considerations of efficiency will continue to take precedence.

METAXIDES:

In many cases users are suboptimizing. At the same time, we as users, complain bitterly when we have to rewrite programs. When we had our discussion on data independence, at no point was it mentioned that the longevity of programs and the need for being able to have complex mappings to get data independence can to a large extent be avoided by sound data base design. Sound data base design means seeing the world in a fashion which reflects that world reasonably well. If I mix together a state and something else and call it some fancy crossbred name, I may be in trouble because there is no counterpart entity in the real world; this may inhibit adding entities that relate to states but not to the crossbred entity that doesn't really exist in the real world.

TURNER:

Several questions have been raised during the data independence discussion on the number of mappings required. It isn't clear that one mapping or two mappings or ten mappings are required. I wonder if the panel would like to discuss this.

ENGLES:

The assertion is made that we want to change the program's definition independently from the storage definitions. The question is how can that be done unless there is something in the middle to provide a level of indirection? In other words, two mappings are required.

BACHMAN:

Ed Sibley presented a picture which showed the Super-schema which I think is the missing piece in this background.

The concept of the missing piece is reasonably well described in the GUIDE-SHARE requirements document. Some other work being done by the ANSI/X3/SPARC Data Base Committee has suggested that the thing in the middle, the super-schema or conceptual schema has value. The most permanent thing we can think of is the conceptual view of our business. We are not required to have any sense of efficiency in terms of how fast it runs because it's a view of the business. When relating this to a logical structure and then to a physical structure, the data-base administrator must continually make decisions to optimize and at times reoptimize the content of his actual files. This can be viewed as something a restructuring program would do from time to time to bring the data base back into efficient balance. Starting from the other end, I'd like a user to be able to have particular subviews of the conceptual world. He would not like this view disturbed any more than possible because that would obsolete a procedure or some way he chose to manipulate that file. The conceptual schema is essentially like the rock of Gibraltar, something to anchor things to. It had some reasonably long term stability, certainly more than weeks, and hopefully months and years with add-ons. The two-step mapping allows you to change either one and just rebuild the single mapping which takes you from the virtual to the conceptual view, then to the logical, and from logical to storage. Compiling techniques of course could combine the two maps into a single map which might have efficiency measures of interest.

GRAVES:

I don't see where this super-schema is used during any computerized execution. It seems that the application program is accessing the sub-schema as it did in the DBTG proposal, and the DBMS in doing the physical access is going through the schema as it did in the DBTG proposal. The super-schema hung out there in mid-air. All I can see is that it might be the data dictionary. Is that where it is used or does it really exist inside a computer?

BACHMAN:

It might be very similar to the dictionary in the sense of being a complete comprehension of your business, things future, present, and past. The reason that you want it in the computer is that you'd like to use it as a pivotal point against which you edit both schemas, the logical schema (the schema in the DBTG sense) and the virtual schema (sub-schema). If you look at what happens in the virtualization of records, fields, or sets, they must all be derivable from the conceptual schema.

This ensures that all the pieces come from some place that has been described. This gives you some editing rules to determine what you can put into a sub-schema. You can't call for fields which are not known or derived them from the fields that are not described in the conceptual schema.

The only place that the schema or sub-schema have come into focus is in the DBTG report. This intermediate concept is not well focused in that report. Eventually it will split the schema into two parts, one which represents how I logically want to store it, the other, how I conceptually think of it.

OCHEL:

Mr. Bachman indicated that the super-schema will remain reasonably firm over some long period of time. If the super-schema is indeed a model of the real world, then we've got to expect it to change just as often as the schema or the sub-schema. There may be some necessity for it as a mapping function, but I don't really believe we can expect it to be any less changeable than any of the other levels.

METAXIDES:

It is essential in an overall data base management system to have what I call an extremely flexible cataloguing scheme or data dictionary facility. It would be possible to build direct connections from that cataloguing scheme, which incidentally would not only describe data, but programs and the interrelationship between programs and data. Indeed at some future time it may become practicable to generate a schema directly from the catalogue.

MELTZER:

Charlie Bachman recalled the GUIDE-SHARE Requirements document that specified descriptors of data in storage, the stored record descriptors, and all concepts of storage attributes were located there and there only. There was a logical record descriptor of what the application program would see. In between would be a relatively stable set of things called entity record descriptors. Maybe the names are out of fashion. The concept remains. This intermediate phase is conceptual, used for mapping purposes only, and would have no storage attributes at all. The author of the document did not envision a large single monolithic schema but rather a collection of these entity records which could overlap in concept or in scope. As the world gradually changed some of those descriptors would be re-written, and many programs would be unaffected by a particular change. In fact where the mappings were precisely definable, reversible, and with symmetric operations available, it may be that two of these entity records totally overlapping but with slightly different shapes would exist. Some programs would be affected by the changes. For example, if your program is budgeting under the real estate division and the ownership of the furniture is no longer in the real estate division, that program is going to change. Those programs that are affected by the real world change will have to be rewritten. The fact that we didn't have to rewrite the whole schema permitted all those programs not affected by the specific change to remain unaffected. I strongly suspect that the general concepts are agreed upon. Everyone likes the idea of levels of indirection now. Everybody likes the idea that storage ought to be declared in storage descriptors. I suggest that we consider very carefully the degree of isolation each gives us

and the degree of isolation each of those levels makes available.

BACHMAN:

There is an activity right now, the American National Standard/X3/SPARC/Data Base Committee, approximately twelve people who have been working with some diligence over the last nine months, trying to address this very carefully. They have been asked by SPARC to determine whether there is something pertinent to standardization at the present time. There are many potential interfaces, and this group is trying to establish interfaces where it might be appropriate to standardize, in such a way that the industry could live with it, the user of the products of the industry would be happy with it, and that it would survive for ten or fifteen years.

JARDINE:

It's interesting to speculate what happens to that kind of activity as storage devices begin to encompass logical relations among data items. What we know now as the hardware interface, that is, the stored data description, is going to move up substantially because it will become convenient to do things to hardware that we now do at some other interface level. What does that do to the kind of standard that one might write? It is not at all clear that the increasing capabilities of hardware, such as a data machine, would necessarily move uniformly through the same interfaces as have been standardized.

METAXIDES:

If our architecture is based on what we see of the real world rather than on the particular tools that we have at the present time, then the replacement of those tools with new ones will have little effect except to make more possible what we are now talking about but not achieving very satisfactorily.

MAIRET:

I would address the comments to Mr. Ochel. It has been my experience in data processing, working for a producer of farm machinery, that the entitites, things, reality, the objects that would be described in the super-schema, are stable. My company has customers, has products, has parts, has people, has money. I suggest that if you do away with any one of those resources, there will be many more considerations than just data processing and changing of programs.

OCHEL:

That is certainly true. It, of course, depends on what we are going to put into the super-schema that reflects the real world. You add new products, new relationships with respect to that product, who it relates to, and so on. It's the relationships involved, rather than the things themselves, that are so changeable. It's the relationships that we are dealing with in a data management environment. Although there are many stable relationships, there are also many unstable relationships. That was the point that I was trying to make.

BACHMAN:

In a sense what Chuck (Mairet) said is even more general than what he stated. I suspect if you go to the next farm equipment manufacturer you will find customers, products, dollars, and people. If you jump across to someone who makes only automobiles, they have customers, products, etc. A business is remarkably alike from one manufacturing company to the next, from one bank to the next, from one insurance company to the next. We will really make some progress when we begin to see what the real business is and not our somewhat limited ability to model it in hardware today.

Data Base Management Systems, D. A. Jardine (ed.).

PANEL DISCUSSION NO. 2.
SOCIAL AND MANAGERIAL ASPECTS OF DATA BASE SYSTEMS

Panel: I. E. Berg
J. A. Gosden
D. P. Moehrke
H. L. Morgan
P. L. Nichols
P. Norden
E. H. Sibley
J. A. Turner, Moderator

NORDEN:

The previous panel talked increasingly of modelling the real world through various intermediate layers of transformations and mappings. An important question is, how much ought to be in the Data Base and how much ought to be in the application program. If we allow a world view that we've got something called reality, there are successive layers of models and images and perception of what is happening. That's already a reduction in detail, variety, and dynamics. We then prepare some input and ultimately machine readable material is obtained.

It's my opinion that the dynamics of what is really going on in the world, should be captured in an applications layer. While freely admitting that in the Data Base structure we must be able to perform the linkages via various levels of schema, I don't believe that all of this can be captured strictly in the machine because of the variety of the half-life, of the cruciality of changes that are occurring outside.

SIBLEY:

There are layers of differences between the relationships of the data as users see them, and the layers of relationships of data as they might very well be structured by a Data Base Administrator. We should keep as much as possible of the data structure or the relationships between the data out of the programs. If data independence is to be realized, I would be very much opposed to allowing much of this structure to be in the application program. The real world is something we find very difficult to model. As somebody mentioned our view of it is going to change, but it is presumably changing in a relatively slow fashion, compared to potentially the way we want to store our data.

MOEHRKE:

There are essentially three parts to information systems development: a data base approach, a structured, modular programming approach, and a generalized systems approach. In the data base approach the programs process virtual records where the mapping is, for the most part, predefined. However, the same data may be materialized in a number of different ways depending on the needs of the program. This has two purposes. The first is isolation against future changes in the data base and in the access methods. The data base strategy is primarily a protection mechanism. The second is that a data base materializes data such that it makes sense to the person using it. Structured programming and systems involves conceptualization of hierarchies of modules, including the data base and the applications. We put things together with building blocks. In conjunction with this, however, we also believe in table-driven software whenever possible, which involves merging data and logic. Thus there is a continuum as you go from the data base through the application logic and out to the user. We have table-driven logic, which is essentially saying the data base will define decision rules. The decision rules are not bound to the program, they're bound at the time the table is created. The result of all this is generalized systems such that we can run

many manufacturing businesses with the same software.

NORDEN:

I did not mean to imply that I wanted the data structures in the applications programs, but I would like to imply an asymmetrical converse. I don't want the dynamics of the relationships of a model of the real world necessarily imbedded in the data base or in the data base management system. It has been estimated that there are roughly 20 basic structures that describe all kinds of business or enterprise operations. I would like to see an archetypical market basket of applications, in the external sense, the way an applications user would understand them. We should assess their intrinsic structure, and dynamics, the statics and the data ingredients that they imply, so as to make some intelligent mappings across the various interfaces. Our repertoire of structures is extremely meager. We talk about flat files, hierarchies or trees, and networks. That seems to be the end of it.

LEFKOVITS:

I would like to put a different dimension on Professor Norden's remark. It is my understanding that the data base we're talking about is a public data base. It is owned by a corporation, by a institution, and really the problem of what is in the data base and what is in the applications program is not complete. We have to bring in the user's own private data that either may augment what is in the data base or may contradict what is in the data base. There is an issue of individual doing things as individuals with data, as they would like to deal with it.

TURNER:

We've looked at the problems and needs of data base management systems in terms of defining and managing data. We've discussed the impact of data on society and industry. The mythical beast the Data Base Administrator is expected somehow to tie these two ends together. What are the functions that the Data Base Administrator must perform and who should he be to perform them efficiently for the complete enterprise.

MORGAN:

To borrow a phrase from John Gosden a few years ago, "He should be the Wizard of Oz" and encompass all of the technical, managerial, and political skills needed in an organization. The GUIDE/SHARE document for example describes what role the Data Base Administrator should have. I think it's a good description of the role he plays. As the systems become better, as the technical parts improve, we can remove this role of the Data Base Administrator as responsible for the efficiency of operations. The systems themselves will begin to dynamically reorganize, restructure the data and optimize for efficiency of use. The Data Base Administrator's real role will be to insure at the organizational level that he knows what a particular piece of data means, who is collecting it, where it's coming from, and who is using it. The technical skills required to optimize the data and the standardization function of enforcing discipline will be obviated by better systems. Part of it is 10 or 15 years away. In the meanwhile, we need this "Wizard of Oz" who can handle the technical aspects.

GOSDEN:

It depends upon what tools you have available and what doctrine the corporation as a whole has. We've discovered that in our organization there was no one central focus. We've had to divide the job into different areas depending upon the way the organization operates, and depending upon the automatic pro-

tections that are in the systems themselves. The better systems are, and the better educated people are, the less this will be a police function.

MAIRET:

Part of the problem is the result of talking about a function in terms of a person. If we recognize it as a function, and we've looked at other functions that have been established in our corporation such as financial management and personnel management, we will find that the function pervades all levels of the corporation. We must recognize it not as a person or a set of people, but rather as a function that's going to exist in many places and levels in our corporation. Then our terminology will begin to straighten out and we'll also realize that it is possible.

BASH:

As a Data Base Administrator, I don't own the data, all I do is administer it. A large part of my problem is the political one of trying to ensure that our various departments are talking about the same thing when they say they are, and that our programmers are also. The Administrator really doesn't have a lot of line authority. It's not his job to say how it will be done, but to do it the way that other people want it done. The Data Base Administrator is not going to do much technical work. There is too much technical work for one person. We found that within our company the person who was a technical expert in Data Base design may very well be the same one who is also the Systems Analyst responsible for one of the major functions involved.

SIBLEY:

I believe there's going to be more and more centralization of authority, and that in fact we will see the Data Base Administrator function becoming higher in a corporation, ultimately vice-presidential level. Karl Marx said that accountants are the jackals of capitalism. Maybe Data Base Custodians will be the hyenas of bureaucracy.

NICHOLS:

One of the big problems faced by our Data Base Administrator function is describing what a piece of data is. This may seem like a trivial problem, but it is a monumental problem in my company. We've been working on it for 5 or 6 years, and we don't have any good way of describing what a piece of data is so that everyone in the company can understand it. We can provide aliases so that people can refer to the same thing by different names, but that is not the problem. People don't know what a piece of data is and what it's representing.

GRAVES:

The DBTG report, and its successor the DDLC report, mentions the Data Base Administrator in some detail. It encompasses both the administrative and custodial aspects. The breadth of those responsibilities indicates that it is very definitely a staff and not a single person. There are very diverse requirements for this function, including technical responsibility, at present, for efficiency and storage structure of the data. A staff of people specializing in quite different fields under a single co-ordinator, the Data Base Administrator, is required.

JOYCE:

We have created a new function here which hasn't existed previously. Because of the organization's structure, there is no way to put a staff in sufficient authority to carry out this job properly. Thus, the data adminis-

tration is going to be done fairly badly in many places for several years. Does the panel have any comments or recommendations on how to ensure what proper authority can be given along with the responsibility, so that data administration can be done fairly well as soon as possible?

SIBLEY:

One of my friends said that the Data Base Administrators of the past were the damn good programmers in the shop. Generally they knew what the relationships were with the data and they had a good concept of the dictionary even if it wasn't articulated. They formed the nucleus of the data administration function. I agree that it's probably going to take some time. Data Base systems are going to help push us faster towards it.

MORGAN:

One of the things that's lacking now in software engineering are good standard measurement tools, not just for measuring the efficiency of systems, but how things are used, what values, what data means, how to format all this information in such a way that it can be reasonably presented. Putting out a book that says here is how you'll be a data base administrator at this point in time may create more problems than it solves.

MOEHRKE:

I'd like to think that I could go to my vendor and find out all the answers to all our business problems. We could reduce our management levels considerably if that were the case. The biggest problem that the data base administrator has right now is not technical at all. It's trying to understand what people mean.

NORDEN:

IBM has had experience with installing and actually running two very large information systems, the AAS system in the Data Processing division and the CMIS system in the manufacturing function. I don't believe a single Data Base Administrator was used in either of those. The function evolved because of the needs of some existing business functions. One was the order and sales administration, the other was building computers. The people running them solved primarily questions of when is a computer finished, when is its last test cell done, the last wires on. These are business functions, not technical functions. The biggest problem as to get the agreement of the entire corporation as to the definition and meaning of various states of completion.

WRIGHT:

Every corporation has data base administration at the present time. Almost every corporation has a thing called forms control or record retention, which is actually data base administration. They do the kind of things you've been talking about, describing data, understanding what it is controlling, where it goes, who has it. The problem that we have is to determine what they do and how much of it belongs in data processing, how much in the rest of the corporation. Corporations have been doing data base administration for a long time and they have actually set up formal organizations for it.

TURNER:

I'd like to underscore a point that was made concerning the problems with achieving consistent set of data definitions across a fairly complex enterprise. About a year ago Howard Maynard told me about the difficulty that then Esso International had in describing some hundred or so fields that he has in his system for controlling tankers. The effort required to define those definitions

was not at all trivial. We see this in the University as we try to reach agreement on the data definitions which describe a student.

KIRBY:

Historically back we've been concerned with the value of the output of our data processing and how useful the information will be. We work hard to reduce the cost of data processing. More recently we've started to use data management techniques and the Data Administrator to further reduce the cost of data processing. I don't believe we'll really make a breakthrough until we give the Data Base Administrator responsibility for the data as a resource. There are many techniques that exist today, inventory control techniques, incoming inspection techniques, which stood when they're applied to raw materials or finished goods. These same techniques are just as applicable to the care and feeding of data. Data Base Administrators must be seen as a valuable member of the company who has the responsibility of a resource, and given the kind of standard tools which allow data to be defined once and then used where it's wanted.

METAXIDES:

One of the things that would definitely help to make the data administration function viable, would be some software which would allow documentation to be tied in to the administration of data. The data base management system itself could be used to develop a mechanized method of keeping track of data so that the data administrator has a valuable tool with which all users could communicate. This could take the form of a sophisticated dictionary that would encompass functions extending from the main data element, the definition of data element, records of checking for similar data items and redundancies via text-editing techniques and documentation techniques, confirming that the definitions are the same, finding out what programs are using them, how they're using them, how often they're using them. This would provide much of the information the data administrator needs to do his job.

GOSDEN:

In many companies the functions Kendall Wright talked about are not done at the levels they should be done, and users are as far behind in that as they are in data base administration. We had better understand the problem before we try to automate it. Not every company is as well organized as IBM and some others in record retention. It often is pertinent to only a very small proportion of the fundamental records of the company, those kept for legal or other critical reasons. In reply to Metaxides, it would be very helpful in solving one part of the problem we're talking about.

MORGAN:

In response to Tax Metaxides, I know of one such system already in existence. It is built on a relational data base management system, as a control, but it is attacking only one part of the problem. It doesn't give the Data Base Administrator enough additional power to really help. He needs the organizational clout which was referred to before in order to be able to have to exercise this function in a reasonable way.

SPENCER:

The topic of administration of data makes the basic assumption that the data is in some form to be administered. One of the greatest needs at this point in time is the data base planner. That is, how do we get to the point where we have data to be administered. We have corporate planning staff but rarely if ever do I see a corporate data base planning staff. In many companies the data base

is very restricted, for example a parts data base, a policy data base. Rarely do you go into an office and see on the wall a picture of what the corporate data base will look like. Data processing is often a service function to departments rather than to the corporation. We should do far more planning at a corporate level of how are we going to create a data base. We implement systems without really defining the needs for this data five years from now. Data processing tends to be treated strictly as a service organization rather than as an integral part of the corporate operation.

TURNER:

There are two aspects of planning: one is planning within the sphere of influence of the data processing activities, and the other is total planning of a particular enterprise. Within data processing we are guilty of not doing nearly the planning we should. It is not at all clear in many enterprises that data is a resource, that it is a commodity that must be allocated and controlled, and that its use has to be rationally planned.

BERG:

The question of where the DBA or the group is, the rank he holds, and the clout he has, is more likely to be answered by the career progressions of the people representing this fraternity. I suspect that pressure from below, abstract descriptions of what the job and the function and task could be, attracting management's attention to the potential involved, is probably not the realistic route that will end up having been inscribed by those who will integrate the functions at the upper reaches of the management. It will be people from this background who become top management and as a consequence of their own backgrounds invite the kinds of response that the data base administration can use. Top management in my experience simply hasn't had enough exposure except second, third and fourth hand to what information systems are and should be. Unless some of the people from the data processing tradition are at the level of presidents of corporations, we're not likely to see the demands congeal and jell in any large corporation for the full blown positioning of the Data Base Administrator. One simply didn't get any of the major functions in American business located at the decision-making input levels until somebody from those traditions themselves managed to get into top management. Until we reach that point from the computer tradition, from the computer industry, from the Data Base Administrators you'll find the management doesn't know what questions to put to you. Unless someone knows what kinds of data's available and knows what questions to put, it becomes a residual function, a staff function, an overhead function. It doesn't seem to generate profit, can't be a profit centre, and doesn't offer a product. I would guess that the move into top management will not occur rapidly.

RABIN:

Some companies are now experiencing partial consolidation of product development as well as development of information systems. It's difficult to find management who has experience both in the actual product line and in information systems development. If a decision must be made as to who should be the manager, usually the one who knows about the product, but not about the information system, gets the job.

JARDINE:

There exists a hypothesis that a person's activities can be defined entirely, or sufficiently exactly for practical purposes, by the information to which he has access. Looking at him as an information processor, his job is defined by the inputs and outputs, to oversimplify somewhat. In the average corporation, a person has access, even formally, to a very broad segment of information concerning the position that he currently holds in the enterprise. He has a fairly

wide formal context. The memoranda, the pieces of paper that go across his desk, very often have information that are outside his immediate concern. With the development of large scale data base systems it seems to me possible that the amount of information extraneous to his current activity diminishes. His data is much more focussed and in fact, he may not be able to browse through other information because he is not authorized to do so. In some cases the person's job and his ability to progress may be constrained by the development of large scale information system.

BERG:

Rather than limiting a manager in his informal information flow, the sort of serendipity that occurs, the systems will do far more in limiting him to more relevant information and keeping down the garbage that fills most people's desks. Whether the extraneous data is really relevant and whether the hour he spends thinking about it really help him to do his job, I would question somewhat.

METAXIDES:

Exception reportion has been touted very widely, and yet we see these mounds of papers still being generated. As an analogy, my last car had an oil warning light rather than a gauge. The advertisers were very clever. They had heard of exception reporting and their advertisements said that they were providing a light rather than a gauge, because this was exception reporting and was much easier to handle. Unfortunately, the switch that controlled the light broke, the lamp itself would break from time to time. When that happened, it was necessary to install a gauge to determine what was happening. Possibly exception reporting is a little overstated in that it restricts the amount of information, unless it is very well done.

SIBLEY:

However, I have heard one comment from a manager, to the effect that exception reports are all right provided that you have a solid basis of knowledge. If you are not careful with exception reports, you may react to the exception without being aware of the ordinary good things that are happening.

BERG:

Coming back to Don Jardine's related question, I had the opportunity about six or seven months ago of repossessing all the chronogical files that came from my own office during a two and one-half year period which I was a acamedic management. The bulk of the content of memorandum work were not informational. In the crudest sense, one could call them political. I don't think what is in those memoranda really is much affected by the availability of warning lights or exception reports or steady flow of data available at the push of a terminal button. What is really in the memorandum is the record of something that went wrong. There were errors in judgment that were dealt with. There were excuses to be made. There were explanations and embroideries, so an issue like the one that was mishandled or less than fully handled, could be better handled next time. My conclusion was that most of those memoranda were really wasted because no one knew the systems inferences and the systems implications of most of the enormous volume of material. In two and one-half years, it came to literally thousands of pages of carefully dictated reactions. While one would refocus some of it, one would not eliminate the volume of it. We simply end up turning it a little and making it a bit more informed in the best sense of the word. It doesn't get eliminated, and one doesn't lose power from it. One quick analogy is the collective bargaining agreement. Even if we had full disclosure of management's books, even if there are full agreement on both sides of the bargaining table what all the data meant about overtime, and about fringe benefits, costs, and

output for man hour, and all the relevant statistical parameters, you still don't get away from the bargaining problem of who wants what under what conditions and with what kinds of quid pro quo. The same thing would happen with our data base system with a highly placed figure of the vice-presidential level working closely with management. The negotiations would simply have to continue because you are still making an enormous number of value judgments.

MOEHRKE:

You can draw some analogies in many systems areas with exception reports for quality control. You have to be concerned with noise, and the filtering of noise. Any exception reporting system has the same kind of obligation, and that is why you set it up as an exception report. One reason is to avoid the mounds of data and the other is to filter out noise. You don't report an exception unless it gets beyond some bounds.

RABIN:

The criticism has been made in some papers published a few years ago that a manager might lose contact sitting in his office relying only on information systems, and not doing proper planning. The question is how much does the information system reflect to the world.

BERG:

In the absence of these information systems, in many organizations that I am familiar with, you get the same kind of lack of reality. Decision making is not necessarily informed by a concern with reality. It is very often concerned with the repayment of our debts and obligations, and what is referred to here in different contexts as tradeoffs. The relationship between the decision and the reality that it will influence is quite variable. There are other values informing the judgment process than, for example, the business realities or the competitor realities. There are legal issues that come up which are not adjudicable on the basis of information. They are adjudicable only on the basis of the value judgment. I would not be particularly pessimistic about losing contact with reality as a consequence of having mounds of cross-tabulated or other statistically manipulable data.

MURRAY:

Because of the fact that you had information available, you were able to show that requiring an advanced degree for a certain occupation is not justified because you couldn't prove that the performance of the advanced degree was better. This is a problem with data base systems. There are many inequities in the system that nobody can currently measure because the data isn't available. We are going to create these data base systems and make the data available, and now some executive officer armed with this kind of data can be deadly.

MORGAN:

If the availability of this data is going to make him a better manager, and being more deadly means being a better manager, then that's what we are here for. That is what the systems are supposed to provide.

BERG:

I'm not sure that I would draw the inference from my data that many of you have apparently thought I was implying. If we discover a lot of people's education is unmatched with performance in the organization, that doesn't necessarily lead management to conclude that recruiting strategy should be changed.

It is altogether conceivable that one detects an untapped resource. One starts re-organizing functions and tasks to better utilize the people one has. Instead of changing the admittance requirements of jobs, we can change jobs, more fully exploit the capabilities that are there, add functions to people who are willing to undertake them. If viewed that way, the answer to your question really is that the information does not necessarily point to any given answer. It can help eliminate some of the options, and help sharpen the attention that has to be given to whatever non-parametric judgments have to be made. The scope for dispute can be narrowed and that has to facilitate an improvement in management. It simply means that one is arguing much more concretely over options and alternatives.

TURNER:

In Tom Steel's presentation there was some very large numbers displayed. There was a figure of 250,000 data bases and a very, very, large sum of money involved in data base related functions. Does the panel have observations, comments or opinions on those figures, and some of their implications.

STEEL:

There was a Delphi study done by Guide about three years ago which identified a large number of expected applications of computer systems, and estimates were derived as to when they were expected to be in place. I went through that list, noted ones that appeared to me to involve a data base management system, and used two standard deviations as the estimate of when the applications would occur. I then looked at the number of locations where that kind of application made some sense. I had fairly accurate figures for the United States, and I doubled it to account for the rest of the world.

JARDINE:

I can't question the basis of Tom Steel's estimate of the number of data bases but I have some feel for the size of the data bases. He quoted figures up to ten to the 14th characters per data base. There won't be many ten-to-the-fourteenth character data bases in the time-frame considered; but there will be possibly one. There will certainly be a number of ten-to-the-twelfth character data bases and ten-to-the-tenth to ten-to-the-eleventh character data bases will be extremely common. This can be derived, for instance, by looking at the five largest insurance companies, taking the number of reels of tape in their vault, and dividing by two to get rid of redundancies. You may have to multiply that by two in order to get it back into the data base including all the relationships and accessing information. Estimates made this way fall into the above ranges. IBM's AAS system is currently of the order of ten-to-the-eleventh characters. That is one of the larger data bases at the moment. At least that order of magnitude will be relatively common in the next five to eight years.

TURNER:

Assuming the general validity of Tom Steel's forecast, what type of an effect might this have on ourselves, on our environment in general; what are some of the implications of systems of these sizes.

STEEL:

One very obvious implication is, if we don't do it right its going to be extremely expensive. If you are out by a factor of two in terms of building these things right, that amounts to an awful lot of money. Sometimes we are out by an order of magnitude in our estimates of how much it should have cost versus how much it actually cost. If you take my hundred billion dollars and multiply that by ten, that is a lot of money.

WIEDERHOLD:

How do the costs compare with the gross national product at that time?

STEEL:

I think that it is about 3%.

NORDEN:

The estimates made by Tom Steel are not so much what will happen but rather the effect of a process that is already occurring. More and more organizations and organizational fragments are beginning to avail themselves of real time conversation capabilities encompassing more and more information. In essence, the boundaries of the things with which we concern ourselves in a machineable way, and how they are inter-related, are increasing. With each round of success we begin to get more greedy or more confident that we can reliably and reasonably economically handle more of this information storage and flow in a competent way as a tool and as a way to do business. The growth in that direction is the outgrowth of different management styles and the way we begin to design enterprises. The fact that we can begin to talk about the design of enterprises, will allow us to design in a wholelistic interactive sense a series of groupings of departments, groupings of factories, distribution systems, feedback of customer satisfaction and dis-satisfaction, as entities with their intermediate buffers , their cross impact on each other. This is more than centralization. What we design is no longer just a machine, but enterprises and groups and clusters of enterprises up to certain segments of the society as a whole.

METAXIDES:

The rate at which we progress towards this will depend on the rate at which we are successful in building better and better data base management systems and improving their cost performance. I don't feel at all frightened by the figures in that I believe every age, every year, the data bases that are in existence will be those that meet the needs of the world. Today, mechanized or not, there are as many data bases in existence as are necessary. Data bases are not anything particularly associated with computers. They have been in existence since the beginning of time. We should not be intimidated by figures the size of these because the process is evolutionary. No organization will attempt overnight to put all of its activities on a data base. We should be building data base management systems that allow an organization to start at some given point and to expand and grow.

BERG:

You have to look at these data base projections in conjunction with at least two other pieces of data. One is the statistics he showed us on the exponential increase in the demand of services. The other is the intriguing table about the rest of the world, where the GNP's were trivial, where the per capita GNP's were almost non-existent even in the year 1995. There was a per capita GNP in the undeveloped countries of about $340 a year, which compares with a per capita GNP in the United States now of approximately ten thousand dollars. When one starts looking at those figures, the expansion of data bases is not nearly so frightening or intimidating, assuming that it is done intelligently and we're not simply collecting data for the sake of collecting data. What you are doing is designed ultimately to facilitate the productive capacity of some kind of entity, a corporation, university, a library, or reader in a library. If data base systems could facilitate our getting plumbing to India such that we could eliminate dysentery, it might be much more important than training physicians in American hospitals. I would infer from those statistics, exponential increase

in the numbers of the data bases simply reflect a larger portion of the world population trying to deal with a larger number of problems than are dealt with in Europe, the United States and people of the Soviet Union in 1973. I would also hazard a guess that the figures on services are specific and interesting in connection with public service. A fantastic amount of the data base increase, the sheer numbers of data bases that we have, will have to do with social policies, health, welfare education. Viewed that way all those million and millions of data bases really almost look inadequate to the informational needs that we have just on the scientific front in solving the problem of productivity across the globe. One would hope well-managed entities, whether they be hospitals, corporations, universities or Indian villages, would be better managed because of these things, and more people will be served with more and more goods and services than is presently the case. Better information, more quickly available to public policy making, can hardly but serve the world.

COHN:

I am not intimidated by the growth and size of data bases. I am not intimidated by their potential uses. However, unless we provide legislative safeguards, I am going to feel intimidated by the potential intimidating misuse of those data bases.

STEEL:

There is one additional parameter of interest. The number of systems is a reasonably conservative estimate; the dollar figure is also quite conservative. That dollar figure amounts to about 8 man-years per data base for design, development, installation, and maintenance.

MAIRET:

The money that we now spend as companies processing data is far far bigger that that. If we do a proper job of implementing good information or data technology and are able to manage it properly, we will substantially decrease the cost to our companies and to our society of processing data. I am not talking about computer costs, because there is a far bigger cost that I think that we can deal with very effectively.

Data Base Management Systems, D. A. Jardine (ed.).

ATTENDEES

Adamson, R.W.
Central Intelligence Agency,
Anderson, L.A.
Eastman Kodak Company,
Bachman, C.W.
Honeywell Information Systems Inc.,
Bash, C.
Dow Chemical Company,
Bedekar, S.
Burroughs Corporation,
Berg, I.E. Jr.
Columbia University,
Black, J.
Sperry Research Center,
Brown, D.
Fireman's Fund American,
Brown, E.C.
MRI Systems Corporation,
Bushell, M.
IBM Corporation,
Campbell, B.
IBM Corporation,
Canning, R.G.
Canning Publications,
Chinlund, T.
Columbia University Computing Centre,
Codd, E.F.
IBM Corporation,
Cohen, L.J.
Performance Development Corporation,
Cohn, L.
IBM Corporation,
Coleman, C.T.
The Johns Hopkins University,
Cooke, V.
Kaiser Foundation Hospitals,
Cox, R.
General Electric Company,
Date, C.
IBM Corporation,
Delport, L.
University of Leuven,
Drager, L.
Northern States Power Company,
Emerson, E.J.
Burndy Corporation,
Emery, J.
Wharton School of Business,
Engles, R.
IBM Corporation,
Fraccola, L.
Honeywell Information Systems Inc.
Gallitano, J.M.
Education Information Services,
Gibb, K.R.
Bell Laboratories,
Gilliam, J.
B.F. Goodrich,
Goodrich, R.
Brighman Young University,
Gosden, J.A.
Equitable Life Assurance Society,
Grace, A.G.
Travellers Insurance Company,
Graves, R.
The National Cash Register Company,
Grotenhuis, F.
N.V. Philips Gloeilampenfabrieken,
Hadcock, W.J.
A.O. Smith Corporation,
Hagerth, S.A.
Fireman's Fund American,
Harper, L.
IBM Corporation,
Hartley, P.A.
Sperry-UNIVAC,
Heinonen, J.
Computer Sciences Corporation,
Herre, H.F.
Fireman's Fund American,
Hill, P.
IBM Corporation,
Hill, W.
Sun Life Assurance Co. of Canada,
Hobbs, L.C.
Hobbs and Associates,
Hoffman, W.
E.I. Dupont de Nemours and Co.,
Hughes, D.
CINCOM Systems Inc.,
Jackson, W.M.
United States Government,
Jardine, D.A.
Queen's University,
Joyce, J.D.
General Motors Research Laboratory,
Kamerman, A.
IBM Corporation,
Kerr, R.
Insurance Company of North America,
Kirby, R.J.
Northern Electric,
Kirshenbaum, F.
Equitable Life Assurance Society,
Kunecke, H.
Boeing Computer Systems
Kurz, R.C.
Southern Railway Company,

Lefkovits, H.C.
Honeywell Information Systems Inc.,
Lightfoot, J.
Rockwell International,
Lowenthal, E.I.
MRI Systems Corporation,
Metaxides, A.
Bell Laboratories Raritan River,
Mairet, C.E.
Deere and Company,
Marx, G.
IBM Corporation,
Maynard, H.S.
EXXON International,
McGraw, E.
Raytheon Data Systems,
McNeil, D.H.
Shared Medical Systems,
Meltzer, H.S.
IBM Corporation,
Mercer, W.
Eli Lilly and Company,
Mickey, M.
Jos. Schlitz Brewing Company,
Moehrke, D.P.
A.O. Smith Corporation,
Morgan, H.L.
University of Pennsylvania,
Mullany, J.
Wayne State University,
Murray, J.T.
General Motors Technical Center,
Nichols, P.L.
Ontario Hydro,
Nies, T.
CINCOM Systems Inc.,
Norden, P.
Columbia University,
Ochel, I.A.
MRI Systems Corporation,
Ohayon, S.
Xerox Data Systems,
Olson, R.M.
Control Data Corporation,
O'Reilly, M.
Bell-Northern Research,
Pierce, J.
Xerox Data Systems,
Plagman, B.
Federal Reserve Bank of New York,
Plitt, L.A.
Control Data Corporation,
Puerling, B.W.
Bell Telephone Laboratories,
Rabin, J.
Western Electric,
Ramm, M.
Fireman's Fund American,
Raver, N.
IBM Corporation,
Reinstein, H.C.
IBM Corporation,
Richley, T.
CINCOM Systems Inc.,
Rivest, E.L
General Electric Company,
Rustin, R.
Chase Manhattan Bank,
Saunders, M.
Control Data Canada, Ltd.,
Schein, M.
IBM Corporation,
Schymik, W.S.
IBM Corporation,
Scott, E.D.
National Cash Register Company,
Sevick, K.C.
University of Toronto,
Sheehan, D.
Equitable Life Assurance Society,
Sibley, E.H.
National Bureau of Standards,
Simmons, W.
CODASYL,
Smith, A.P.
IBM Corporation,
Smith, D.M.
ESSO Math. and Systems, Inc.,
Smith, L.
Bankers Trust Company,
Spillum, M.P.
Employers Insurance of Wausau,
Steel, T.B., Jr.
Equitable Life Assurance Society,
Sternich, H.
The Mitre Corporation,
Stockhausen, P.F.
Wisconsin Telephone Company,
Surtees, R.A.
The National Cash Register Company,
Tani, P.Y.
IBM Corporation,
Tellier, H.
IBM Corporation,
Trask, S.
Sun Life Assurance Co. of Canada,
Traver, C.H.
Stanford University,
Turner, J.A.
Columbia University
Uhrbach, H.
Data Base Design, Inc.,
von Gohren, G.L.
The Pillsbury Company,
Wiederhold, G.
Stanford University,
Worley, W. Jr.,
IBM Corporation
Wright, K.R.
IBM Corporation.

REFERENCES

_____Editor, (1970) a software package produced by Group Operations, Inc., Washington, D.C.

American Bankers Association (April 1971) "Monetary and Payments System Planning Committee: Executive Report" ABA.

Ash, W.L. and Sibley, E.H. (1968) "TRAMP: An Interpretive Associative Processor with Deductive Capabilities". Proc ACM National Conf. p. 143.

Bachman, C.W. (1969) "Data Structure Diagrams". Data Base 1, #2.

Bachman, C.W. (1973) "Set Concepts for Data Structure", Encyclopedia of Computer Science, Auerbach. (scheduled for publication in 1973).

Bachman, C.W. (1973(a)) "The Evolution of Data Structures", Proceedings of The NORDATA Conference, Copenhagen, Denmark.

Bachman, C.W. (1973(b)) "The Programmer as Navigator", CACM 16, p. 653.

Bell, D. (1973) "The Coming of Post-Industrial Society", Basic Books, New York.

Byrnes, Carolyn J. and Steig, Donald B. (1969) "File Management Systems: A Current Summary:, Datamation, October.

Steig, Donald B. (1972) "File Management Systems Revisited", Datamation, October.

Childs, D.L. (1968) "Description of a Set-Theoretic Data Structure", FJCC, p. 557.

CODASYL COBOL Data Base Task Group Report (April 1971) ACM.

CODASYL DEVELOPMENT COMMITTEE (1962) An Information Algebra: Phase I Report, CACM 5, 5, p. 190.

CODASYL SYSTEMS COMMITTEE (1969) A Survey of Generalized Data Base Management Systems, (available from NTIS).

CODASYL SYSTEMS COMMITTEE (1971) Feature Analysis of Generalized Data Base Management Systems. Published by ACM.

CODASYL SDDTTG (1972) "An Approach to Stored Data Definitions and Translation", SIGFIDET Workshop Proceedings, p. 13.

Codd, E.F. (1970) "A Relational Model of Data for Large Shared Data Banks" CACM 13, 6, p. 377.

Crean, P.A. and Lavallee, P.A. (Fall 1972) ARCHIVE File Management System, XDS Users' Group Proceedings.

Crean, P.A. and Lavallee, P.A. (Spring 1973) ARCHIVE: Tape File Management System, XDS Users' Group Proceedings.

Dearden, J. (1972) "MIS is a Mirage", Harvard Bus. Rev., Jan-Feb.

Dodd, G.G. (1966) "APL - A Language for Associative Data Handling in PL/I", FJCC 29, AFIPS Press

Emery, James C. (1971) Cost Benefit Analysis of Information Systems. The Society for Management Information Systems.

Feistel, H. (1973) "Cryptography and Computer Privacy", Scientific American 228:5 p. 15.

Feldman, J.A. and Rovner, P.D. (1969) "An ALGOL-based Associative Language", CACM 12, 8, p. 439.

Gosden, J. and Raichelson, E. (1969) The New Role of Management Information Systems, the MITRE Corp. MTP-332(AD-691-834).

Gosden, J.A. (1969) Software Compatibility: what was promised, what we have, what we need. AFIPS Conference Proc. 33FJCC, p. 81.

Gosden, J.A. (1969(a)) Report to X3 on Data Definition Languages, Journal of ACM SIGFIDET 1, 2, p. 14.

Gosden, J.A. (1970) Mistress - a design concept for a real-time system environment for maintenance and testing. The Equitable Life Assurance Society N.Y. TSG-70-D-1005.

Gosden, J.A. (1972) The Conceptual Requirements for a Management Information Data Bank, Information Processing 71, North Holland, p. 861.

Gosden, J.A. (1972(a)) The Making of a Management Information Data Base, Computer Decisions 4.5 p. 20.

Griffiss, G.L. (July 1971) A practical application of a generalized data entry and validation system for use in a mixed real time and batch environment. The Computer Bulletin, p. 254.

GUIDE-SHARE Data Base Requirements Group (1970) "Data-Base Systems Requirements".

Hanold T. (1972) "An Executive View of MIS, Datamation, November, p. 65

Harrell, C. Jr. (1972) "Maintaining a Healthy Data Base", Business Automation, February, p. 16.

Honeywell Information Systems, Series 600/6000 Integrated Data Store Reference Manual, Order No. CPB-1565, Wellesley, Mass.

IBM (May 1971) The Time Automated Grind System (TAG): Sales and Systems Guide GY20-0358-1.

Japan Computer Usage Development Institute (May 1972) "The Plan for Information Society - a National Goal Toward Year 2000", Computerization Committee Final Report.

Joint Utilities Project, CMS Action Group (July 1971) Data Management Systems Requirements.

Kahn, D. (1967) "The Codebreakers", Macmillan, New York.

Kahn, H. and Bruce-Briggs, B. (1972) "Things to Come", MacMillan, New York.

Kahn, H. and Wiener, A. (1967) "The Year 2000", MacMillan, New York.

Lavallee, P.A. Ohayon, S., and Sauvain, R.W. (Fall 1972) Non-Procedural Access to DMS Data Bases, XDS Users' Group Proceedings.

Lynch, H.H. (1969) ADS: A Technique in System Documentation, Database 1, 1, p. 6.

McGee, W.C. (1972) "Some current issues in data description", SIGFIDET Workshop Proceedings, p. 1.

Mealy, G.H. (1967) "Another Look at Data" FJCC p. 525.

The Mitre Corp. (1973) Data Managemen Systems Catalog, Report #MTP-139.

Myers D.H. (1962) A time automated technique for the design of information systems, IBM Systems Research Institute, New York.

National Cash Register Company (1967) Accurately defined systems.

Ohayon, S. (Fall 1972) IDDP: A low-level interactive, self-contained System for Interrogation and Update of a DMS Data Base, XDS Users' Group Proceedings.

Perlmutter, H.V. (1971) "The Multinational Corporation: A New Kind of Institution", Diebold Professional Paper.

Plagman, B.K. and Altshulter, G.P. (1972) "A Data Dictionary/Directory System within the Context of an Integrated Data Base", AFIPS Conference Proceedings 41.

Plagman, B.K. and Altshuler, G.P. (1972(a)) "User/System Interface Within the Context of an Integrated Corporate Data Base", Diebold Europe.

Plath, W.J. (1972) "Restricted English as a User Language", Proceedings of GUIDE 33.

Polis, R.T., Morresi, A.G. and Lemmon, A.Z. (1969) Proposed capability Design specification for an automated input processing system MITRE Corp., MTR-5073.

Raichelson, E. (1972) "A concept paper on Host vs. Own Data Manipulation Languages in Military Information Systems", SIGFIDET Workshop Proceedings, p. 67, Published by ACM.

Sayani, H.H. (1973) "A Decision Model for Restart and Recovery from errors in information processing systems". PhD Dissertation, University of Michigan, Ann Arbor, Mich.

Senko, M.E. et al, (1973) "Data Structures and Accessing in Data-Base Systems" IBM Systems Journal, 12, 1, p. 30.

Severance, D.G. (1972) "Some Generalized Modeling Structures for Use in the Design of file organizations", PhD Dissertation, University of Michigan, Ann Arbor, Mich.

Shagnasty, T.B. (1973) "One Man's Data Base", SHARE Secretary Distribution (SSD-234).

Sibley, E.H. and Taylor, R.W. "A Data Definition and Mapping Language", CACM in publication.

Steel, T.B. Jr. (1964) "Beginnings of a Theory of Information Handling", CACM 7, 2, p. 97.
Summers, J.K. and Bennett, E.M. (1967) Aesop A, Final Report, a prototype on-line interactive information control system. The Third Congress of Information Systems Science Technology, Thompson, Washington, D.C.
Teichroew, D. (1972) "A survey of languages for stating requirements for computer-based Information Systems", FJCC 41, p. 1203.
Welke, L. (1972) "A Review of File Management Systems", Datamation, Oct., p. 97.
Westin, A.F. and Baker, M.A. (1972) "Data Banks in a Free Society", Quadrangle, New York.
Whitney, V. Kevin M. (1972) "A Relational Data Management System (RDMS)" General Motors Research Publication-1293.
Wylie, K. (1971) "Summary of Results; GUIDE/IBM Study of Advanced Applications," GUIDE International and IBM.
Xerox Data Systems (1971) Xerox Universal Time-Sharing System (UTS) Manual 900907C.
Xerox Data Systems (1972) DMS Reference Manual Version C00.